META-RESEARCH Workshop on Advanced Treebanking

Advanced Treebanking 2012

LREC2012 Workshop
22 May 2012
Istanbul, Turkey

Editors

Jan Hajič	Charles University in Prague, Czech Republic
Koenraad De Smedt	University of Bergen, Norway
Marko Tadić	University of Zagreb, Faculty of Humanities and Social Sciences, Croatia
António Branco	University of Lisbon, Portugal

Workshop Organizing Committee

Jan Hajič	Charles University in Prague, Czech Republic
Koenraad De Smedt	University of Bergen, Norway
Marko Tadić	University of Zagreb, Faculty of Humanities and Social Sciences, Croatia
António Branco	University of Lisbon, Portugal

Workshop Programme Committee

António Branco	University of Lisbon, Portugal
Sabine Buchholz	Toshiba Research Europe, Cambridge, UK
Khalid Choukri	ELRA/ELDA, Paris, France
Silvie Cinková	Charles University in Prague, Czech Republic
Dan Cristea	University of Iaşi, Romania
Koenraad De Smedt	University of Bergen, Norway
Rebecca Dridan	University of Oslo, Norway
Nancy Ide	Vassar College, New York, USA
Valia Kordoni	DFKI, Berlin, Germany
Sandra Kuebler	Indiana University, Bloomington, USA
Krister Lindén	University of Helsinki, Finland
Paul Meurer	Uni Computing/Uni Research, Bergen, Norway
Adam Meyers	New York University, USA
Joakim Nivre	University of Uppsala, Sweden
Stephan Oepen	University of Oslo, Norway
Marco Passarotti	Catholic University of the Sacred Heart, Milan, It.
Eiríkur Rögnvaldsson	University of Reykjavik, Iceland
Victoria Rosén	University of Bergen, Norway
Mária Šimková	Slovak Academy of Sciences, Bratislava, Slovakia
Barbora Vidová Hladká	Charles University in Prague, Czech Republic
Fei Xia	University of Washington, USA
Daniel Zeman	Charles University in Prague, Czech Republic

Workshop Programme

Tuesday, 22 May 2012

09:00 – 09:10 Jan Hajič
Welcome and Introduction to the Workshop

09:10 – 10:30 **Oral presentations – Session I**
co-chaired by Jan Hajič and Koenraad De Smedt

09:10 – 09:35 Tom Vanallemeersch
Parser-independent Semantic Tree Alignment

09:35 – 10:00 Philippe Blache and Stéphane Rauzy
Hybridization and Treebank Enrichment with Constraint-Based Representations

10:00 – 10:25 Bruno Guillaume and Guy Perrier
Semantic Annotation of the French Treebank with Modular Graph Rewriting

10:25 – 10:30 Jan Hajič
Introduction to the Poster Session

10:30 – 11:30 **Coffee break with poster presentations**

Victoria Rosén, Koenraad De Smedt, Paul Meurer and Helge Dyvik
An Open Infrastructure for Advanced Treebanking (with demo)

Oleg Kapanadze
A German-Georgian Parallel Treebank Project

Tomáš Jelínek, Vladimír Petkevič, Alexandr Rosen and Hana Skoumalová
Czech Treebanking Unlimited

João Silva, António Branco, Sérgio Castro and Francisco Costa
Deep, Consistent and also Useful: Extracting Vistas from Deep Corpora for Shallower Tasks

11:30 – 13:00 **Oral presentations and invited speech – Session II**
co-chaired by Marko Tadić and António Branco

11:30 – 11:45 Hans Uszkoreit, coordinator of META-NET (invited speech)
High-Quality Research, New Language Resources and their Sharing

11:45 – 12:10 Rajesh Bhatt and Fei Xia
Challenges in Converting between Treebanks: a Case Study from the HUTB

12:10 – 12:35 Maytham Alabbas and Allan Ramsay
Arabic Treebank: from Phrase-Structure Trees to Dependency Trees

12:35 – 13:00 Gyri Losnegaard, Gunn Inger Lyse, Martha Thunes, Victoria Rosén, Koenraad De Smedt, Helge Dyvik and Paul Meurer
What We Have Learned from Sofie: Extending Lexical and Grammatical Coverage in an LFG Parsebank

13:00 **End of Workshop** (start of lunch break)

Author Index

Table of Contents

Preface

Many R&D projects and research groups are creating, standardizing, converting and/or using treebanks, thereby often tackling the same issues and reinventing methods and tools. While a fair amount of treebanks have been produced in recent years, it is still a challenge for researchers and developers to reuse treebanks in suitable formats for new purposes. Standardization of interchange formats, conversion and adaptation to different purposes, exploration with suitable tools, long term archiving and cataloguing, and other issues still require significant efforts.

In this spirit, the present workshop has been conceived by four projects, namely T4ME, META-NORD, CESAR and META4U, which under the META-NET umbrella project strive to make many treebanks and other language resources and tools available for R&D. It is hoped that the workshop will contribute to innovative insights that will promote development, dissemination, use and reuse of treebanks in the future.

Thirteen papers were submitted to the workshop, of which ten were accepted for presentation at this half-day workshop. Six were selected for oral presentation while four were selected for poster presentation. We thank all our reviewers for their constructive evaluation of the papers.

Jan Hajič
Koenraad De Smedt
Marko Tadić
António Branco

Parser-independent Semantic Tree Alignment

Tom Vanallemeersch

Centre for Computational Linguistics, KU Leuven (Belgium)
tallem@ccl.kuleuven.be

Abstract

We describe an approach for training a semantic role labeler through cross-lingual projection between different types of parse trees, with the purpose of enhancing tree alignment on the level of syntactic translation divergences. After applying an existing semantic role labeler to parse trees in a resource-rich language (English), we partially project the semantic information to the parse trees of the corresponding target sentences (specifically in Dutch), based on word alignment. After this precision-oriented projection, we apply a method for training a semantic role labeler which consists in determining a large set of features describing target predicates and predicate-role connections, independently from the type of tree annotation (phrase structure or dependencies). These features describe tree paths starting at or connecting nodes. The semantic role labeling method does not require any knowledge of the parser nor manual intervention. We evaluated the performance of the cross-lingual projection and semantic role labeling using an English parser assigning PropBank labels and Dutch manually annotated parses, and are currently studying ways to use the predicted semantic information for enhancing tree alignment.

1. Introduction

This paper deals with the question whether we can use semantic predicates and roles for automatically aligning syntactically divergent parts in parse trees of parallel sentences. We will focus on the aspect of providing trees with such semantic information, and give a general idea on how to use the information for alignment. We will look specifically into the language pair English-Dutch.

Semantic predicates and roles operate at the level of meaning rather than syntactic structure. For instance, at the syntactic level, *he breaks the window* expresses a relation between a verb on one hand and its subject and object on the other hand, whereas the semantic level shows a relation between a predicate (the verb), its "agent" (*he*) and its "theme" (*the window*). The semantic relation between predicate and theme can also be expressed by a passive syntactic construction, in which *the window* is the subject, or by a construction without direct object (*the window breaks*). Semantic predicates and roles operating at a more abstract level than syntactic structure, we investigate whether they allow to find correspondences between nodes of a parse tree in one language (henceforth called the source parse tree) and the parse tree of its translation equivalent (the target parse tree[1]) that are difficult to detect by merely looking at similar syntactic structures in the parse trees.

Apart from the example of divergence mentioned above, there are numerous other types of divergences, such as translation of a verb by a noun (*they adapt the techniques* vs *de aanpassing van de technieken*, i.e. 'the adaptation of the techniques'). Divergences may lead to a different structure of the source and target parse tree. Their structure can also differ due to the type of parse tree (dependency-based versus constituency-based parsing). Figure 1 shows an example of a translation divergence involving an active vs a passive construction. If we know that the subject of the English verb *safeguard* has "agent" as role, and that the *obj1* node is the "agent" of the Dutch verb, we find the correspondence.

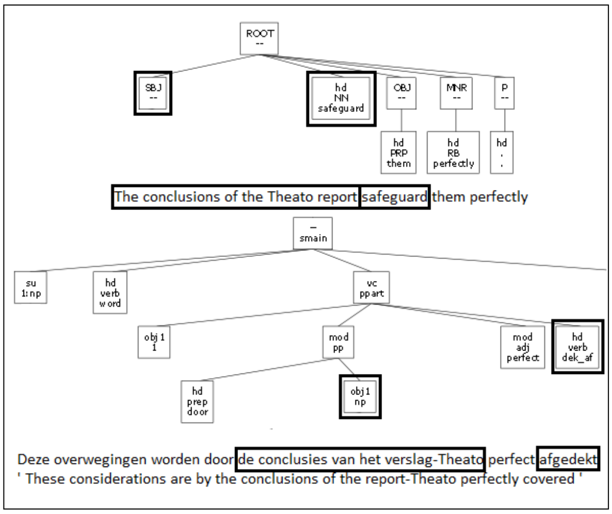

ROOT
--
SBJ
--
hd
NN
safeguard
OBJ
--
MNR
--
P
--
hd
PRP
them
hd
RB
perfectly
hd
.
.
The conclusions of the Theato report safeguard them perfectly
–
smain
su
1:np
hd
verb
word
vc
ppart
obj1
1
mod
pp
mod
adj
perfect
hd
verb
dek_af
hd
prep
door
obj1
np
Deze overwegingen worden door de conclusies van het verslag-Theato perfect afgedekt
' These considerations are by the conclusions of the report-Theato perfectly covered '

Figure 1: Translation divergence: the English SBJ node corresponds to the *obj1* node in the Dutch passive construction (nodes below *SBJ*, *su* and *obj1* are not shown).

2. Semantic frameworks

Semantic predicates and roles can be described as a way to answer the question *who did what to whom, and how, when and where?* (Palmer et al., 2010). In his seminal paper, Fillmore (1968) defines the notion of "deep cases", such as *agentive* and *instrumental*. Several semantic frameworks have been devised, and databases implementing them. VerbNet (Kipper-Schuler, 2005), based on the diathetic verb alternations of Levin (1993), associates verbs to roles such as Agent and Theme. FrameNet (Baker et al., 1998) describes prototypical situations that are linguistically expressed through frames. For instance, the frame

[1] When using the terms "source" and "target", we abstract from the actual translation direction, i.e. the source language may not necessarily be the language in which the sentences were originally written.

STATEMENT is “triggered” by the verbs *state*, *speak*, *report* etc. A third example of a framework is PropBank (Palmer et al., 2005), which describes the proto-agents and proto-patients of specific verbs, a categorization which is based on the theory of Dowty (1991). The argument which shows the highest number of proto-agent (agent-like) properties is labeled *ARG0*, the one which shows the highest number of proto-patient (patient-like) properties is labeled *ARG1*. Verbs may also have a *ARG2*, *ARG3* or *ARG4* argument; the interpretation of these arguments is strongly linked to the verb itself (e.g. ARG2 is the recipient of the verb *lease*). The modifiers, like *ARGM-TMP* (temporal modifier), do not depend on the meaning of the verb.

As PropBank was specifically devised in order to make the creation of a semantic database practical and to allow an automatic semantic role labeler to be trained on it, we will experiment with this semantic framework for evaluating the semantic labeling approach described in this paper, and with the complementary NomBank framework (Meyers et al., 2004). The latter describes nominal predicates (deverbal nouns like *recognition*, or nouns unrelated to verbs) using the same roles as PropBank.

3. Strategies for semantic annotation

In this section, we will tackle the question of how to provide the source and target parse tree with semantic information. We assume there exists a semantic role labeler for the source language. If there is no labeler (of the same type) for the target language, there are several alternatives. One is to create manually annotated sentences and train a semantic role labeler on them. A second alternative is to apply cross-lingual projection of semantic predicates and roles using word alignment, see e.g. Padó (2007), and to induce additional semantic predicates and roles in the target language through manually crafted rules involving knowledge of the target parser, e.g. guessing the predicate in the target sentence if only source roles were projected (Vanallemeersch, 2010). A third alternative is to apply cross-lingual projection and train a labeler on the projected information.

Each of the alternatives mentioned has some disadvantages. The first alternative is time intensive. The second alternative requires in-depth knowledge of the target parser, and can only induce predicate-role relations if there exists a partial projection. The third alternative requires a manual selection of features from the target parser for building a semantic role labeler. In order to tackle the dependency on parsers and manual intervention, we devised a variant of the third method, which automatically selects features from the target parser to train a semantic role labeler. This way, the method not only remains independent from the target parser, but also makes it independent from the source parser: if we transfer the manual or automatic annotations in source parses to parses of another type, we can train a labeler on them.

In the following sections, we describe our procedure for crosslingual projection and for training a semantic role labeler.

4. Crosslingual projection

The idea of crosslingual projection consists in the transfer of information of some kind from sentences in a resource-rich language (typically English) to their equivalents in another language, using alignment links, such as word alignment. This allows for inducing a preliminary resource for the second language in a reasonable amount of time. Projection was originally applied to syntactic annotations. Later on, it was also applied to semantic information. Padó applied it to an English-German corpus, transferring FrameNet annotations to the German part through a word alignment established by GIZA++ (Och and Ney, 2003).

In our approach, we enforce the following projection condition: if the GIZA++ intersective word alignment (which is highly precise) contains a link between a source tree node with a semantic predicate and a target tree node X, and if the word alignment allows to establish an “exclusive” match between one of the roles of the source predicate and a target tree node Y, consider X the semantic predicate in the target tree, and Y the semantic role of X. An exclusive match between two nodes implies that all word alignment links starting at one of the nodes end up in the other node. Successful projection leads to a target parse node annotated as predicate, and one or more nodes annotated as its roles. For instance, the predicate *safeguarded* and its role *conclusions of the report Theato* in Figure 1 can be projected to the target tree. As we combine the precision of exclusive matching with that of GIZA++ intersective alignment, we are quite confident about the correctness of the predicate-role annotations in the target trees. Obviously, the target parse tree may contain additional predicate-role relations which are not annotated by the projection, because of the precision of the projection and the sparseness of the word alignment. This sparseness can have several causes, such as the syntactic divergences between the source and the target sentences, the fact that they are paraphrases, etc.

In order to apply machine learning techniques for training a semantic role labeler, we also project negative information. This allows for learning both from positive and negative examples. We determine which nodes in the source tree are not predicates (all nodes not annotated as predicates in the source tree), and which nodes are not roles (all nodes not annotated as roles in the source tree). We project these non-predicates and non-roles to the target parse through intersective word alignment and exclusive matching. All nodes in the target parse tree that are not annotated through projection of positive or negative information are ignored by the machine learning process described below; we do not know whether they are predicates or roles. In the Dutch tree in Figure 1, we see many non-predicates (the *su* node, the *obj* node, . . .) and non-roles (the *dek_af* node, the *su* node, . . .).

5. Features for semantic role labeler

We now turn to a description of our approach for training a semantic role labeler. Semantic role labeling typically involves several steps, one for predicate identification, one for identifying their roles and one for identifying the type of the roles. For this purpose, classifier models are trained on semantically annotated parses; together, they perform the task of semantic role labeling. For instance, if the predicate was identified, a classifier model for identifying its roles is typically trained on features such as the phrase type and the parse tree path (Palmer et al., 2010). In order to establish the list of features, some knowledge of the parser is required. Parser-dependent restrictions may be imposed on the identification of roles. For instance, they may only be identified within the smallest clause containing the predicate node.

The approach we propose here aims at the automatic determination of features for classifier models, and as such at independence from specific parsers. Our approach considers all properties of paths which start at specific nodes, or connect two nodes, as features. The features have the following form, `<direction>` being either up (↑) or down (↓): `<feat=val>[<direction><feat=val>]*`

For instance, the following features are determined for the Dutch predicate node in Figure 1 (note that the visualization tool[2] we used only shows values, not features like *rel*, and that we apply a threshold on the number of feature-value pairs in the path property):

- `rel=hd`
- `rel=hd` ↑ `rel=vc`
- `rel=hd` ↑ `cat=ppart`
- ...

The following features are determined for the Dutch predicate-role path in the example in Figure 1:

- `rel=hd` ↑ `rel=vc` ↓ `rel=mod` ↓ `rel=obj1`
- `rel=hd` ↑ `cat=ppart` ↓ `rel=mod` ↓ `rel=obj1`
- ...

We look for all features which occur in the set of positive and negative examples of nodes and paths. For instance, when building a classifier model for identifying predicate nodes, we determine the features of predicate nodes and non-predicate nodes. Each example becomes an instance in the training set for classification, and is described using the features. The target class of the instances in the training set is *true* (i.e. the instance is a predicate node) or *false* (i.e. the instance is a non-predicate node). When building a classifier model for identifying predicate-role paths, we determine the features of paths between predicate nodes and their roles, and between predicate nodes and their non-roles. The latter include all nodes of which we know with certainty they are not a role of the predicate: (1) roles of other predicates, (2) predicates, (3) non-predicates, and (4) non-roles.

Even for a restricted number of nodes and paths, the number of features determined may be significant. We therefore store the features and their frequency in a trie. Each feature is a key in the trie, and the labels leading to the key have the form `<direction>?<feat=val>`). By adding features with an increasing number of feature-value pairs, and filtering out features occurring less than a threshold, we keep the size of the trie and the time needed to create it manageable. The trie moreover allows to store the features of several classifier models in a compact way (we store information for multiple models on the trie nodes).

Using the above mechanism to determine features, we can build several types of classifier models, each with their own features:

1. a model for identifying paths between predicates and their roles; its classes are `<role type 1>`, `<role type 2>`, ..., `none` (the last class is used for non-roles of the predicate node)
2. for each specific predicate observed (e.g. *come*), a model for identifying paths between the predicate and its roles; its classes are `<role type 1>`, ...
3. a model for identifying predicate nodes: its classes are *true* and *false*
4. a model for identifying role nodes: its classes are *true* and *false*
5. ...

The predictions of the second model get priority over that of the first one (the first model acts as a fallback in case there is too little information for a specific predicate).

6. Evaluation

We tested the crosslingual projection using a sample of 500 English-Dutch sentence pairs from Europarl (Koehn, 2005). We parsed the English sentences using the system of Johansson and Nugues (2008) which simultaneously performs syntactic and semantic parsing of English sentences, and assigns PropBank and NomBank roles. We only retained NomBank predicates which are linked to a PropBank roleset (deverbal nouns), and omitted modifiers like *ARGM-TMP*. We parsed the Dutch sentences using Alpino (Bouma et al., 2000). The crosslingual projection was performed using GIZA++ intersective alignment. 49% of the predicates were projected, and 69% of the roles of the projected predicates[3]. A manual assessment of 50 sentences indicated a precision of 94% for predicate projections and 92% for role projections.

In order to test the performance of the semantic role labeling approach, we first performed a test with optimal input for the training step (gold standard). We used a subset of 300 Dutch sentences from the SoNaR project (Schuurman et al., 2010), which were parsed using Alpino and manually annotated with PropBank roles.[4] We automatically determined features for the two first models mentioned in section 5. Modifiers like *ARGM-TMP* were ignored. The features were written to files in the ARFF format used by machine learning programs. We trained the models with a

[2] http://rug-compling.github.com/dact

[3] Crosslingual projection using 500 English-French sentence pairs from Europarl and a parser trained on the French treebank using MaltParser (Candito et al., 2010; Abeillé et al., 2003; Nivre et al., 2007) lead to a higher percentage for predicates (63%) but a lower percentage for the roles of the projected predicates (53%).

[4] The consortium of the project provided us with those parses.

support vector machine using the Weka toolkit[5]. An example of a high-weight feature in the first model is the following:[6] `pos=verb` ↑ `cat=smain` ↓ `rel=su`. We then created test ARFF files for the sentences used as training material. These files describe links between two nodes which have to be classified. The two trained models were applied to the test ARFF files in order to get a class prediction for the links. We did not carry out predicate identification for this test (we assumed the predicates of the sentences to be known beforehand), hence ignored any links not involving the known predicates. The prediction for the remaining links was evaluated as follows:

1. If there was a prediction from the second model (i.e. there is a specific classifier for the predicate in the link), and both the role node and role type predicted are equal to that of the gold standard, the prediction was considered correct. If the prediction involved the same role node but another role type than the gold standard, or a role node not present in the gold standard, the prediction was considered incorrect.

2. If there was no prediction from the second model, the same conditions as above are applied for the prediction of the first model.

For the recall calculation, we kept track of the predicate-role links in the gold standard which did not match a prediction on both the role and role type. The first row in Table 1 shows the precision, recall and F-score of the predictions.

We subsequently tested the semantic role labeling approach by training on automatic annotations, i.e. crosslingual projections. We took a sample of 600 English-Dutch sentence pairs from the DPC corpus (Macken et al., 2007) and performed crosslingual projection in the same fashion as described above for the Europarl corpus. We trained the two first models mentioned in section 5. on the predicates and roles in the Dutch parse trees. We then created test ARFF files for the gold standard of SoNaR, and applied the two models in order to get predictions and evaluate them against the manual annotations in the gold standard. The second row in Table 1 shows the performance. One of the reasons that the models trained on the projection perform worse consists in a lack of information for creating predicate-specific classifiers. For instance, the subject of *komen* ('to come') is labeled as A0 in the projection-based predictions instead of A1 as no classifier model for *komen* was built. However, some types of information which are valid in a more general fashion are captured both by gold standard-based and projection-based models, e.g. the fact that subjects of passive constructions are the A1 of the predicate.

We are currently studying how to use the predicted information for tree alignment. Consider the following sentence pair:

- English: *introduced in the autumn of 2004 , this procedure simplifies the deposit of ...*

Table 1: Statistics of classifier models

	Precision	Recall	F-score
Gold standard models	0.77	0.60	0.67
Projection models	0.67	0.50	0.57

- Dutch: *deze vereenvoudigde procedure werd in de herfst van 2004 ingevoerd en vergemakkelijkt de afgifte van ...* ('this simplified procedure was in the autumn of 2004 introduced and makes-easier the deposit of ...')

The English parser establishes a relation between the predicate *simplifies* and the roles *this procedure* (A0) and *the deposit of ...* (A1). Our semantic role labeling approach predicts a relation between the predicate *vergemakkelijkt* and the roles *deze vereenvoudigde procedure* (A0) and *de afgifte van ...* (A1). As GIZA++ intersective alignment does not contain a link between *simplifies* and *vergemakkelijkt*, the English predicate and role were not projected. However, the word alignment allows to link *this procedure* with *deze vereenvoudigde procedure* and *the deposit of ...* with *de afgifte van ....* This way, we have sufficient information to align both the predicates and the roles.

Another example:

- English: *a systematic survey of the various national regulations*

- Dutch *de nationale regelgeving moet worden onderzocht* ('the national regulation must be examined')

The English parser establishes a relation between *survey* (nominal predicate) and *the various national regulations* (A1). Our semantic role labeling approach predicts a relation between the predicate *onderzocht* and *de nationale regelgeving* (A1). GIZA++ intersective alignment contains a link between *survey* and *onderzocht*; based on the predicted information, we can link their roles *the various national regulations* and *de nationale regelgeving*.

7. Conclusions and future research

We have discussed a method for cross-lingual projection and parser-independent semantic role labeling. The projection method is precision-based. Training the semantic role labeler does not require any knowledge of the parser nor manual intervention. Training classifier models on parses with manual annotation of PropBank labels leads to a higher performance than training on projected information. The question remains to be seen in how far this is a question of the size of training data; in this respect, projection data have the advantage of being much more obvious to produce than manual annotations.

We are currently training and testing several types of classifier models. We also include modifiers such as *ARGM-TMP* in the training, and explore ways to use the predicted information for enhancing tree alignment on the level of syntactic divergences. A part of the DPC corpus has been provided with a manual word alignment. We will use this part

[5] http://www.cs.waikato.ac.nz/ml/weka

[6] This is a path between the verb of the main clause and its subject.

as a gold standard for evaluating tree alignment, and compare our tree alignment results with that of two other tree alignment systems, Lingua-Align (Tiedemann and Kotzé, 2009) and Sub-Tree Aligner (Zhechev and Way, 2008).
As future research, we envisage including predicted semantic information as a feature in Lingua-Align, and using the mechanism for automatic feature determination from nodes and paths for other purposes than semantic role labeling.

8. References

A. Abeillé, L. Clément, and F. Toussenel. 2003. Building a Treebank for French. In *Treebanks : Building and Using Parsed Corpora*, pages 165–188. Springer.

C.F. Baker, C.J. Fillmore, and J.B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98, pages 86–90, Stroudsburg, PA, USA. Association for Computational Linguistics.

G. Bouma, G. Van Noord, and R. Malouf. 2000. Alpino: Wide-coverage Computational Analysis of Dutch. In *Proceedings of CLIN 2000*, pages 45–59.

M. Candito, B. Crabbé, and P. Denis. 2010. Statistical French Dependency Parsing: Treebank Conversion and First Results. In *Proceedings of LREC-2010*, pages 1840–1847, Valletta, Malta.

D.R. Dowty. 1991. Thematic Proto-Roles and Argument Selection. *Language*, 67(3):547–619.

C.J. Fillmore. 1968. The Case for Case. In Emmon W. Bach and Robert T. Harms, editors, *Universals in Linguistic Theory*, pages 1–88. Holt, Rinehart and Winston, New York.

R. Johansson and P. Nugues. 2008. Dependency-based Semantic Role Labeling of PropBank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Proceedings of EMNLP '08, pages 69–78, Stroudsburg, PA, USA. Association for Computational Linguistics.

K. Kipper-Schuler. 2005. *VerbNet: a Broad-coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, Philadelphia, PA, USA.

Ph. Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of MT Summit*, pages 79–86.

B. Levin. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. The University of Chicago Press.

L. Macken, J. Truskina, H. Paulussen, L. Rura, P. Desmet, and W. Vandeweghe. 2007. Dutch Parallel Corpus. A multilingual annotated corpus. In *On-line Proceedings of Corpus Linguistics 2007*, Birmingham, United Kingdom.

A. Meyers, R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman. 2004. Annotating Noun Argument Structure for NomBank. In *Proceedings of LREC-2004*, pages 803–806, Lisbon, Portugal.

J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and E. Marsi. 2007. MaltParser: a Language-independent System for Data-driven Dependency Parsing. *Natural Language Engineering*, 13(2):95–135.

J.S. Och and H. Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29:19–51.

S. Padó. 2007. *Cross-Lingual Annotation Projection Models for Role-Semantic Information*. Ph.D. thesis.

M. Palmer, D. Gildea, and P. Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31:71–106.

M. Palmer, D. Gildea, and N. Xue. 2010. *Semantic Role Labeling*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

I. Schuurman, V. Hoste, and P. Monachesi. 2010. Interacting Semantic Layers of Annotation in SoNaR, a Reference Corpus of Contemporary Written Dutch. In *Proceedings of LREC-2010*, Valletta, Malta. European Language Resources Association (ELRA).

J. Tiedemann and G. Kotzé. 2009. A Discriminative Approach to Tree Alignment. In *Proceedings of the Workshop on Natural Language Processing Methods and Corpora in Translation, Lexicography, and Language Learning*, pages 33–39, Borovets, Bulgaria. Association for Computational Linguistics.

T. Vanallemeersch. 2010. Tree Alignment through Semantic Role Annotation Projection. In *Proceedings of Workshop on Annotation and Exploitation of Parallel Corpora (AEPC)*, pages 73–82, Tartu, Estonia.

V. Zhechev and A. Way. 2008. Automatic Generation of Parallel Treebanks. In *COLING 2008*, pages 1105–1112, Manchester, United Kingdom.

Hybridization and Treebank Enrichment with Constraint-Based Representations

Philippe Blache, Stéphane Rauzy

Aix-Marseille Université & CNRS
LPL, 5 Avenue Pasteur, 13100 Aix-en-Provence
{blache;rauzy}@lpl-aix.fr

Abstract

We present in this paper a method for hybridizing constituency treebanks with constraint-based descriptions and enrich them with an evaluation of sentence grammaticality. Such information is calculated thanks to a two-steps technique consisting in : (1) constraint grammar induction from the source treebank and (2) constraint evaluation for all sentences, on top of which a grammaticality index is calculated. This method is theoretically-neutral and language independent. Because of the precision of the encoded information, such enrichment is helpful in different perspectives, for example when designing psycholinguistics experiments such as comprehension or reading difficulty.

1. Introduction

Besides syntactic description and NLP tools development, treebanks also play an important role in more specific perspectives such as ambiguity resolution (Koller and Thater, 2010), discourse analysis (Webber, 2009) or spoken language description (Wouden et al., 2002). More and more frequently, treebanks are used for interdisciplinary studies, in particular in psycholinguistics, bridging the gap between experimental studies (e.g. eye-tracking) and linguistic resource description (Keller, 2004; Demberg and Keller, 2008; Tomanek et al., 2010).

These different works share the fact that they rely on different types of information (morphology, syntax, semantics, or prosody), encoded at different levels (word forms, categories, sentences) and possibly incomplete. The problem is that a classical representation in terms of constituency hierarchy is not the right level of description for these new tasks (parsing spoken languages, building difficulty models, etc.), in particular because failing in representing partial structures, ill-formed constructions, etc.

Developing treebanks with a finer granularity of syntactic description is then necessary. Constraint-based representations are well equipped in such perspective: constraints can be very precise, possibly not connected to each others and may bring together different levels of representation. Treebanks bearing such precise information would then be of great help. Unfortunately, constraint parsers are of great complexity and often suffer from over-generation.

We propose in this paper to bypass this problem: instead of building constraint-based treebanks from scratch, we propose an hybridization technique building the constraint-based representation starting from a constituency-based one. Knowing syntactic structure (the original tree) dramatically reduces the search space required when building the constraint representation. The interest is that this technique is entirely automatic. It consists first in inducing a constraint grammar from the source treebank and then to build the constraint-based representation thanks to a set of constraint solvers exploiting this grammar. This technique, on top of being efficient, is *generic*: it is independent from any linguistic formalism as well as from the language: it can be applied to any constituency treebank. Moreover, other kinds of information derived from the constraint-based representation, such as grammaticality level, can also enrich the structure, opening the way to new applications, for example in psycholinguistics.

After a brief presentation of the main characteristics of the constraint-based representation, the grammar induction process is described. It consists in gathering all the possible realizations of the different categories of the corpus. The result is a large context-free grammar on top of which the constraint grammar is generated. The third section presents the hybridization mechanism which build the constraint treebank starting from the constituency. The application of this process to the *French Treebank* (Abeillé, 2003) is described and some results are discussed. The last section describes a treebank enrichment: the evaluation of the grammaticality completes the description of the different categories realized in the treebank.

2. Constraint-Based Syntactic Representation

Phrase-structure representations use a unique explicit relation, hierarchy, that encode constituency information. All other information such as linear order, headedness, dependency, etc. are implicit. On the opposite, constraint-based representations encode explicitly all these relations, making it possible to verify their level of satisfaction and to evaluate precisely the structure grammaticality. Such syntactic representation syntax has been experimented in different projects (Blache, 2005). In terms of parsing, the technique consists in considering relations as *constraints*, satisfaction being the core of the parsing process. We propose to represent syntactic information by means of six different types of constraints that describe phrase-level constituents:

- *Linearity*: linear precedence relations between the constituents of a phrase
- *Obligation*: possible heads of a phrase

- *Dependency*: dependency relations between the constituents
- *Uniqueness*: constituents that cannot be repeated in a phrase
- *Requirement*: mandatory cooccurrence between categories
- *Exclusion*: cooccurrence restriction between categories

A complete grammar can be represented by means of such constraints: each phrase-level category is described by a set of constraints between constituents. Parsing an input comes to evaluating for each category the set of constraints that describes it. Concretely, for a given category and a given set of constituents, the mechanism consists in satisfying all constraints, for example verifying that within the set of constituents, the head is realized (*obligation*) or that *linearity* constraints hold. At the end of the evaluation process, the parser has built a set of evaluated constraints. As a consequence, parsing two different realizations of a same category (in other words two different sets of constituents) will result in different sets of evaluated constraints (that can possibly be violated). The final set of evaluated constraints for a given input (also called a *characterization*) forms a description graph, as illustrated in figure 1 (we will use in the remaining of the paper examples from the *French Treebank*).

We can see in this graph how constraints represent explicitly the different kinds of syntactic information. In particular, it illustrates the fact that the number of evaluated constraints can be very different from one constituent to another. This property, together with the fact that constraints can be violated, is of central interest because describing precisely the syntactic relations, not only in terms of hierarchy. Such a rich representation makes it possible to *quantify* these two aspects of the syntactic structure: density and quality. We describe in the following a method for enriching constituency treebanks with such information, independently form the language.

3. Constraint Grammar Induction from Constituency Treebanks

Even if some effort have been done in terms of homogenizing the different syntactic annotation schemes (Abeillé, 2003), the encoded information can be very different from one treebank to another. For example functional annotation can be more or less precise or dependent from the chosen formalism (compare for example (Bies et al., 1995), (Abeillé et al., 2003), (Telljohann et al., 2004) or (Böhmová et al., 2003)). Still, constituency treebanks contains by definition, on top of the morpho-syntactic level, the hierarchical structure. We present in this section a procedure acquiring automatically the different constraints corresponding to the implicit grammar of the treebank. The mechanism is based on the analysis of all possible constructions of the different categories which corresponds, in terms of context-free grammars, to the set of the possible right-hand sides of non-terminal categories.

Calculating the construction sets consists for all non-terminal categories *XP* in traversing the treebank and identifying its daughters. The result, noted $RHS(XP)$, is made of ordered subsets of categories.

Let's note in the following $\prec$ the precedence relation between two categories into a construction. The set of constraints is then calculated for each non terminal category *XP* as follows:

- *Constituency*: for each non-terminal category *XP*, its set of constituents, noted *const(XP)*, is the set of categories participating to the constructions in *RHS(XP)*. Let's note that the tagset used in the constraint grammar to be built can be adapted at this stage: categories can be, according to the needs, more precise or at the opposite more general than that of the initial tagset.

- *Linearity*: the precedence table is built in verifying for each category preceding another category into a construction (or a right-hand side) whether this relation is valid throughout the set of constructions

```
∀ $rhs_m \in RHS(XP)$
    **if** $((\exists\ (c_i, c_j) \in rhs_m \mid c_i \prec c_j)$
    **and** $(\nexists\ rhs_n \in RHS(XP) \mid (c_i, c_j) \in rhs_n\ \wedge c_i \prec c_j))$
    **then** add $prec(c_i, c_j)$
```

- *Uniqueness*: the set of categories that cannot be repeated in a right-hand side.

```
∀ $rhs_m \in RHS(XP)$
    $\forall\ (c_i, c_j) \in rhs_m$
        **if** $c_i \neq c_j$ **then** add $uniq(c_i)$
```

- *Requirement*: identification of two categories that cooccur systematically in all constructions of an *XP*.

```
∀ $rhs_m \in RHS(XP)$
    $bool \leftarrow ((c_i \in rhs_m) \wedge (c_j \in rhs_m))$
    **if** $bool$ **then** add $req(c_i, c_j)$
```

- *Exclusion*: when two categories never co-occur in the entire set of constructions, they are supposed to be in exclusion. This is a strong interpretation, that leads to over-generate the number of such constraints. However, it is the only way to identify it automatically.

```
∀ $rhs_m \in RHS(XP)$
    $bool \leftarrow \neg((c_i \in rhs_m) \wedge (c_j \in rhs_m))$
    **if** $bool$ **then** add $excl(c_i, c_j)$
```

Besides this direct acquisition from the treebanks, two other constraint types require explicit formulation:

- *Obligation*: the heads of a phrase. Identified as the minimal set of compulsory constituents. Usually, this set is identified by means of specific sets of rules (cf. (Lin, 1998)). Note that multiple heads are allowed.

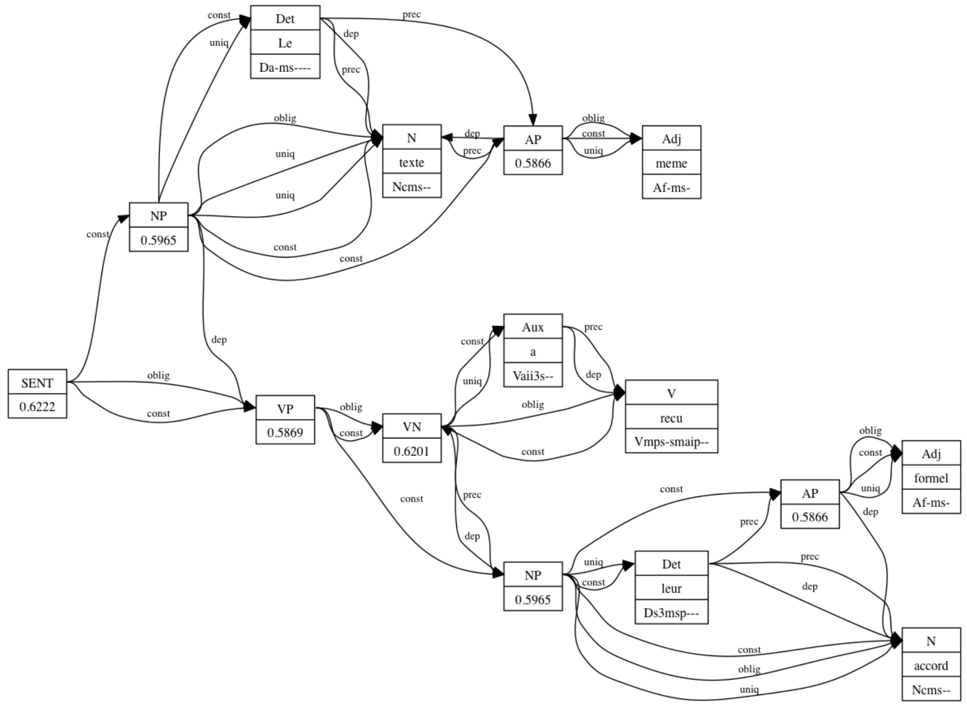


Figure 1: Description graph of the sentence“*The text itself had received their formal agreement*”

- *Dependency*: this constraint typically encodes grammatical relations. When present, they usually encodes complementation, modification or specification. Automatic acquisition in this case is dependent on the treebank and the way such relations are encoded.

The example figure 2 illustrates the acquisition process for the adjectival phrase. We work in this experiment on a subset of the *French Treebank* (Abeillé et al., 2003), made of 134,445 words, manually corrected. The figure 2 presents the list of *AP* constructions in this treebank. The total number of *rhs* is indicated in the right column, the total number of *AP* in the sub-corpus being 2,657. There are 56 different types of *rhs* (for sake of space, only the a subpart is mentioned here). We can note that the 5 most frequent *rhs* types represent 95% of the possible constructions (providing important indication when building probabilistic grammars). It is also interesting to note that in 84% of the cases, the *AP* is formed by a unique constituent, its head. On the opposite, complex constructions, made of more than two constituents are very rare.

In this encoding, without entering into the detail of the tagset, the qualificative adjective is encoded '*Af*', the others being noted '*A-*'. Coordination is indicated at the category level and treated as a feature (for example *NP:COORD*). Finally, punctuation marks correspond to the category *W*.

The extraction of the *AP* constraint system is presented figure 3. The first remark is that this constraint-based representation is very compact. This observation illustrates the fact that, as was observed with the ID/LP formalism in GPSG (Gazdar et al., 1985), a separate encoding of different types of syntactic information makes it possible to factorize a CFG rule-based representation. In fact, our approach systematizes the ID/LP one: we distinguish 6 different types of information where ID/LP takes into account 2 of them. One can see that this representation steps over a level of generalization, thanks to the factorization.

Constituents	Occ.	Constituents	Occ.
Af	1930	A- Ssub	1
A-	302	AdP Af Wm PP Wm PP	1
AdP Af	159	AdP Af Wm NP Wm Ssub	1
Af PP	63	Af Wm Ssub Wm PP	1
Af VPinf	19	AdP Wm AdP Af PP	1
AP Af	17	Af AdP	1
AdP Af Ssub	13	AdP Af AdP	1
AdP Af PP	8	AP Wm Cc AP Wm	1
A- PP	7	NP Af	1
Af Ssub	6	PP Af	1
AP A-	5	Af NP	1
AdP A-	4	AP AdP	1
AdP Af VPinf	3	AdP Wq Af Wq	1
Af PP:COORD	3	A- Wm A-	1
Af NP:COORD	2	AdP Af NP	1
AdP AdP Af	2	AdP Af NP PP VPinf	1
Af PP PP	2	Af Wm NP Wm PP	1

Figure 2: AP realizations in the FTB

Another important remark concerns frequency: it is not necessary to take into account all constructions under a certain frequency threshold. Generally speaking, constructions with at least 2 realizations in the treebank are reasonably representative. By another way, in case of conflict between two constraints, the most frequent one is chosen. It is the case in this example with the linearity constraint between AdP and A: all realizations but one satisfy the constraint $AdP \prec A$. We keep then this constraint in the

const	{AdP, A, VPinf, PP, Ssub, AP, NP}

lin	A $\prec$ {VPinf, Ssub, PP, NP, AP}
	AdP $\prec$ {A, Ssub, PP}
	AP $\prec$ {A, AdP}
	PP $\prec$ {Ssub}

dep	{AdP, VPinf, PP, Ssub, NP} $\rightsquigarrow$ A
uniq	{A, VPinf, Ssub}
oblig	{A}
excl	VPinf $\otimes$ {PP, Ssub}

Figure 3: AP properties

final description.

In our experiment, as presented figure 4, the grammar contains a total of 493 constraints extracted from the treebank (taking into account only constructions with more than one realization). There are important differences between the constructions with respect to the number of constraints as well as their distribution. The *NP*, because of the great variability of its possible realizations, requires a high number of constraints. As for the constraint types, linearity is the most frequent, illustrating the fact that word order in French is highly constrained. It is also interesting to note that the size of the constraint set is not directly dependent from the number of constituents (compare *AdP, Ssub* and *VP*).

The following example, taken from the FTB, illustrates the evaluation of the constraint grammar for the AP "*plus économique que politique* (more economical than politic)":

const	{AdP, A, Ssub}
lin	A $\prec$ Ssub AdP $\prec$ A
dep	{AdP, Ssub} $\rightsquigarrow$ A
uniq	{A, Ssub}
oblig	{A}
excl	VPinf $\otimes$ Ssub

This example shows the interest of a constraint-based description which provides many precise information not directly accessible in a constituency-based representation. We will see in the last section of the paper the importance of such description for different applications.

4. Enriching Treebanks with a Constraint-Based Description

Our treebank enrichment consists in building an hybrid syntactic representation, one (the original) being purely constituency-based, the second being constraint-based. Generally, building a constraint-based representation as described in section 2 is a computationally complex process, highly non-deterministic, in particular due to constraint relaxation. However, the task in our case is to enrich an existing treebank. The problem consists to evaluate the constraint system for each node of the original tree instead of building an entire graph description starting from the initial set of words. Concretely, the process comes to traverse for each sentence its original tree. At each node, the constraint set describing the corresponding category is evaluated thanks to different constraint solvers, presented in the following in terms of set operations.

We note $|E|$ the cardinality of the set E; $\mathcal{C}$ the ordered set of constituents of the category taken into account; $\mathcal{C}_{i..j}$ the sublist of $\mathcal{C}$ between positions i and j; c_i a constituent of $\mathcal{C}$ at position i; n the number of different constituents belonging to $\mathcal{C}$.

Constraints are of two types: those specifying a set (obligation, uniqueness) and those defining relations between sets of categories. In the first case, we note $\mathcal{S}_{cx}$ the set of categories specified by the constraint of type cx. In the second case, we note $\mathcal{L}_{cx}$ and $\mathcal{R}_{cx}$ respectively the left and right parts of the constraint cx.

- *Obligation*: this operation consists in verifying the presence of one of the obligatory categories specified in the obligation constraints. In terms of sets, this means that the intersection between the set of realized constituents $\mathcal{C}$ and the set of categories specified in the obligation constraint:

  $|\mathcal{C} \cap \mathcal{S}_{oblig}| > 0$

- *Linearity*: the verification of the linear precedence constraints consists in verifying that when a category belonging to a left-hand side of a linearity constraint is realized, then no category of the right-hand side can be realized in the sublist of the current category list of constituents preceding it:

  $\forall c_i \in \mathcal{L}_{lin},\ \nexists c_j \in \mathcal{R}_{lin}$ such that $c_j \in \mathcal{C}_{1..k} \wedge c_i \in \mathcal{C}_{k+1..n}$

- *Uniqueness*: this constraints verifies that the specified categories are not repeated, which means that the intersection of the category and the set of realized constituents is not greater than one:

  $\forall c_i \in \mathcal{S}_{uniq}, |c_i \cap \mathcal{C}| \leqslant 1$

- *Requirement*: when one category of a LHS of this constraint is realized, then one of its RHS should too:

  $\forall c_i \in \mathcal{L}_{req} \wedge c_j \in \mathcal{R}_{req},\ c_i \in \mathcal{C} \Rightarrow c_j \in \mathcal{C}$

- *Exclusion*: when one category of a LHS of this constraint is realized, then no category of its RHS should be present:

  $\forall c_i \in \mathcal{L}_{req} \wedge c_j \in \mathcal{R}_{req},\ c_i \in \mathcal{C} \Rightarrow c_j \notin \mathcal{C}$

Thanks to these mechanisms, a constraint-based annotation can be built on top of the constituency structure. Concretely, this mechanism makes it possible to build a parallel treebank as well as an hybrid one. In the first case, two different sets of syntactic annotations are built: one representing the constituency representation, the other the constraint-based one, both of them being aligned at the word level. Another type of representation of the treebank consists in enriching the constituency structure: the description of non-terminal categories (the nodes of the tree) is completed by a set of relation between its constituents (their daughters). The example in figure 6 illustrates this second approach.

Enriching a constituency tree consists in calculating the set of constraints of the FTB constraint grammar that are satisfied for each node of the tree. As presented above, the result consists in a characterization, which is the set of constraints

	AdP	AP	NP	PP	SENT	Sint	Srel	Ssub	VN	VNinf	VNpart	VP	VPinf	VPpart	Total
const	10	7	13	7	8	8	7	10	6	7	7	10	9	8	115
dep	5	6	18	5	3				5	5	6	8			59
exc	1	2	44		2	6	3	3							61
req			6							4	4				14
lin	18	10	36	6	5	4	7	14	11	6	7	24	13	7	165
oblig	1	1	4	1	1	1	1	1	1	3	2	1	1	1	20
uniq	4	3	10	3	3	4	4	1	2	4	5	3	7	6	59
	39	22	131	22	22	23	22	29	25	29	31	46	30	22	493

Figure 4: Distribution of the constraints in the FTB-acquired grammar

```
<category label="SENT" sample_index="0" sentence_index="0:16" node_index="0:16:0">
   <category label="NP" features="NP:SUJ" node_index="0:16:1">
      <category label="Det" features="Da-ms----" node_index="0:16:2" form="Le" lemma="le"/>
      <category label="Noun" features="Ncms--" node_index="0:16:3" form="texte" lemma="texte"/>
      <category label="AP" features="AP" node_index="0:16:4">
         <category label="Adj" features="Af-ms-" node_index="0:16:5" form="mme" lemma="mme"/>
      </category>
   </category>
   <category label="VP" features="VP" node_index="0:16:6">
      <category label="VN" features="VN" node_index="0:16:7">
         <category label="Aux" features="Vaii3s--" node_index="0:16:8" form="avait" lemma="avoir"/>
         <category label="Verb" features="Vmps-smaip--" node_index="0:16:9" form="reu" lemma="recevoir"/>
      </category>
      <category label="NP" features="NP:OBJ" node_index="0:16:10">
      <category label="Det" features="Ds3msp---" node_index="0:16:11" form="leur" lemma="leur"/>
      <category label="Noun" features="Ncms--" node_index="0:16:12" form="accord" lemma="accord"/>
      <category label="AP" features="AP" node_index="0:16:13">
         <category label="Adj" features="Af-ms-" node_index="0:16:14" form="formel" lemma="formel"/>
      </category>
   </category>
</category>
<category label="Pct" features="Wd" node_index="0:16:15" form="." lemma="."/>
```

Figure 5: Example of a tree in the FrenchTreeBank

that can be evaluated for this specific realization. The example figure 6 shows the enrichment for the subject *NP* . In this encoding, the characterization is a set of constraints encoded by the elements < property >. In this representation, all constraints are encoded as relations between two nodes. In the case of set constraints (for example uniqueness constraints that specifies the categories that cannot be repeated), the corresponding evaluated constraint is encoded as a relation between the node and the category. Each element contains 4 attributes: the type of the corresponding constraint, its source and target and the result of its satisfaction (true or false). In this example, the characterization represents linearity *Det* $\prec$ *N, Det* $\prec$ *AP*, dependencies between *Det, AP* and *N*, etc. As mentioned above, the interest of such representation lies in the fact that it offers a precise description of the different syntactic properties.

Moreover, each constraint is evaluated independently and can be satisfied or possibly violated. This means that such representation can also encode non-grammatical, partial or ill-formed constructions. Let's imagine for example that in our example, the noun would precede the determiner. The only difference in the characterization would then be the value of the attribute sat, set to *false* in the corresponding constraint:

```
<prop type="lin" srce="0:16:2" tget="0:16:3" sat="f"/>
```

This characteristic is interesting when describing specific data such as second-language acquisition, spoken language, pathological productions, etc.

Table 1 recaps some figures of the FTB sub-treebank described above and the application to the enrichment procedure. The first table indicates the number of categories observed in the treebank. As already underlined, *NP* is by far the most frequent category, followed by *PP*. Moreover, as mentioned above, *NP* has a great variability of realizations, in comparison to other categories, which has also consequences on the distribution of the constraint types. The second table indicates the total number of evaluated constraints for the treebank, indicated per type. In this case too, we can observe a great difference in the distribution at a first glance. However, this aspect mainly comes from the frequency of the *NP* that uses a lot of exclusion and uniqueness constraints.

SENT	1 471
NP	8 127
AP	2 632
VP	2 550
VN	2 628
PP	4 124
AdP	1 733
Srel	508
Ssub	476
Sint	352
VPinf	917
VPpart	618
VNinf	863
VNpart	616

lin	27 367
obl	32 602
dep	21 971
exc	89 293
req	11 022
uni	38 007

Table 1: Number of constraints by category and type

It is interesting to have a closer look at the distribution of the different evaluated constraints by category. The results for the FTB sub-treebank are presented in figure 7. Note that the number of constraints in the grammar is not directly correlated with the number of evaluated constraint (which is expected) but also to the frequency of the category: *PP* is a very frequent category with an

```
<category label="NP" features="NP:SUJ" node_index="0:16:1">
   <category label="Det" features="Da-ms----" node_index="0:16:2" form="Le" lemma="le"/>
   <category label="Noun" features="Ncms--" node_index="0:16:3" form="texte" lemma="texte"/>
   <category label="AP" features="AP" node_index="0:16:4">
      <category label="Adj" features="Af-ms-" node_index="0:16:5" form="mme" lemma="mme"/>
   </category>
   <characterization>
      <property type="lin" source="0:16:2" target="0:16:3" sat="p"/>
      <property type="lin" source="0:16:2" target="0:16:4" sat="p"/>
      <property type="req" source="0:16:3" target="0:16:2" sat="p"/>
      <property type="dep" source="0:16:2" target="0:16:3" sat="p"/>
      <property type="dep" source="0:16:4" target="0:16:3" sat="p"/>
      <property type="oblig" source="0:16:1" target="0:16:3" sat="p"/>
      <property type="uniq" source="0:16:1" target="0:16:2" sat="p"/>
   </characterization>
</category>
```

Figure 6: Example of a FTB enriched tree for the NP

average number of constraints in the grammar, but represents only 7% of the number of evaluated constraints. This figure is to be compared to that of the *SENT* category, which represents 31% of the total. However, and this very clear when comparing with the *NP*, the frequency of exclusion and uniqueness constraints, which is highly variable, mainly explains this observation.

More interestingly, the respective role of constraint types for each category can be measured when putting together these different information. In particular, the frequency of the constraint type in the treebank for a given category has to be balanced with its frequency in the grammar: a constraint type very frequent in the treebank and with few instance in the grammar will play a more important role in the syntactic description. In other words, the respective weights of constraint types for each category can be automatically evaluated thanks to these figures. We propose the following formula:

$$weight(cx) = \frac{freq_{\mathrm{tbank}}(cx)}{freq_{\mathrm{gram}}(cx)} \qquad (1)$$

Figure 8 presents the application of this measure to the FTB. We can observe for example that for the *NP*, even though the evaluation of the exclusion constraint is much more frequent than others, its relative importance is balanced with respect to others types such as requirement or linearity, which is expected from a syntactic point of view.

5. Enriching Treebanks with Grammaticality Information

We present in this section the application of a grammaticality evaluation technique making use of to the constraint-based enrichment presented in the previous section. Such information, as mentioned in the introduction, can be of great help in particular for psycholinguistics experiments.

One of the characteristics of our constraint-based representation is that it is possible to quantify the number of constraints and their relative importance. This evaluation have been described in (Blache et al., 2006) and relies on the study of the set of evaluated constraints. The method proposes different scoring terms on top of which a grammaticality index is calculated.

We note in the following N_c^+ the amount of constraints satisfied by the constituent *c*, N_c^- the constraints violated, N_c^+, and E_c the total number of constraints that received an evaluation (i.e. $N_c^+ + N_c^-$). We note T_c the total amount of constraints (evaluated or not) specifying the category c. Constraints being weighted, we note W_c^+ (respectively W_c^-) the sum of the weights assigned to the constraints satisfied (respectively violated) by the constituent c. The different terms are calculated as follows:

- *Satisfaction/Violation Ratio*: SR_c (resp. VR_c) is the number of satisfied constraints (resp. violated) divided by the number of evaluated constraints:
$$SR_c = \frac{N_c^+}{E_c} \qquad VR_c = \frac{N_c^-}{E_c}$$
- *Completeness Index*: number of evaluated constraints divided by the total number of constraints for the category c: $CI_c = \frac{E_c}{T_c}$
- *Quality Index*: distribution of the weights of satisfied and violated constraints: $QI_c = \frac{W^+ - W^-}{W^+ + W^-}$
- *Precision Index*: The *Index of Precision* for the constituent c is defined as the following ratio: $PI_c = k \cdot QI_c + l \cdot SR_c + m \cdot CI_c$
  These *adjustment coefficients* (k, l, m) are used as variable parameters for tuning up the model.

Finally, the global *Grammaticality Index* (GI_c) is a function of the previous indexes. It is defined recursively as follows, where c is a constituent and c_i is a nested constituent of c:

$$GI_c = PI_c \cdot \overline{GI_{c_i}} = PI_c \cdot \frac{\sum_{i=1}^{Z_c} GI_{c_i}}{Z_c}$$

Besides the constraint description, each node of the treebank can also be enriched with its grammaticality evaluation as presented in the figure 9.

6. Conclusion

We have presented in this paper a method for enriching constituency treebanks with a constraint-based representation, which offers the interest to propose a very precise representation of syntactic information on top of which automatic grammaticality evaluation can be calculated. Such constraint-based representation has been shown to be adapted to the description of non-canonical input (for example spoken language).

This technique is generic in the sense that it is independent from the source formalism and can be applied to any constituency-based treebank. Grammar induction only depends on the analysis

	AdP	AP	NP	PP	SENT	Sint	Srel	Ssub	VN	Vninf	Vnpart	VP	Vpinf	Vppart	
dep	41	320	7 730	3 871	7 114	0	0	8	854	143	24	1 866	0	0	21 971
exc	4	157	57 232	0	31 820	54	18	8	0	0	0	0	0	0	89 293
req	0	0	6 451	0	4 489	0	0	0	0	72	10	0	0	0	11 022
lin	9	360	9 336	3 895	8 960	0	2	475	965	143	24	2 329	709	160	27 367
obl	1 732	2 562	8 010	3 942	7 073	270	486	463	2 620	863	616	2 523	838	604	32 602
uniq	1 733	2 589	11 586	3 942	10 385	286	506	463	680	144	640	2 642	1 614	797	38 007
	3519	5 988	100 345	15 650	69 841	610	1 012	1 417	5 119	1 365	1 314	9 360	3 161	1 561	22 0262

Figure 7: Distribution of the evaluated constraints in the FTB sub-treebank

	AdP	AP	NP	PP	SENT	Sint	Srel	Ssub	VN	Vninf	Vnpart	VP	Vpinf	Vppart
dep	0.0676	0.2271	0.5050	0.7420	0.4753	-	-	0.1129	0.6340	0.4610	0.0731	0.8971	-	-
excl	0.0330	0.2229	1.5296	-	3.1892	0.2361	0.0949	0.0376	-	-	-	-	-	-
req	-	-	1.2643	-	-	-	-	-	-	0.2901	0.0457	-	-	-
lin	0.0041	0.1460	0.3050	0.6222	0.3592	-	0.0045	0.4789	0.3256	0.3841	0.0626	0.3732	0.3623	0.2196
oblig	14.2734	7.2735	2.3548	3.7783	1.4178	7.0820	7.6838	6.5349	9.7246	4.6364	5.6256	9.7038	5.5672	5.8040
unic	3.5704	2.4501	1.3624	1.2594	0.6939	1.5003	1.6000	6.5349	1.2620	0.5802	2.3379	3.3872	1.5318	1.0941

Figure 8: Weights of the constraint types in the FTB

of the realizations of the different constituents. As a consequence, the different constraints can be generated directly from the original constituency representation. Starting from such grammar, the annotation process itself is entirely automatic. This ensure the consistency of the encoding as well as the reusability of the process. Moreover, it is language independent, the constraint grammar being automatically acquired from the original treebank.

Several works can take advantage from this kind of resources. In particular, grammaticality evaluation makes it possible to compare the different realizations of a same construction as well as quantify its "prototypicity": a high grammatical score usually comes from the fact that the corresponding construction contains redundant information, reinforcing its categorization. This kind of information is useful for example in discourse relations identification, speaker involvement evaluation, etc.

As an example, two on-going experiments rely on such treebanks. First, eye-tracking data are on the process to be acquired for the *French Treebank*, making it possible to look for correlation between grammaticality and difficulty. A second project concerns cross-linguistic study of constraint-based representation, applied to English, Chinese and Arabic. The idea consists here in acquiring a constraint grammar for each of these languages and to compare the description (and the grammaticality) of a same construction through languages.

7. References

A. Abeillé, L. Clément, and Toussenel F. 2003. Building a treebank for french. In A. Abeillé, editor, *Treebanks*, Kluwer, Dordrecht.

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*. Dordrecht: Kluwer.

A. Bies, M. Ferguson, K. Katz, and R. MacIntyre. 1995. Bracketing guidelines for treebank ii style. Department of Computer and Information Science, University of Pennsylvania.

P. Blache, B. Hemforth, and S. Rauzy. 2006. Acceptability prediction by means of grammaticality quantification. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, July.

P. Blache. 2005. Property grammars: A fully constraint-based theory. In H. Christiansen et al., editor, *Constraint Solving and Language Processing*, volume LNAI 3438. Springer.

A. Böhmová, J. Hajič, E. Hajičová, and B. Vidová-Hladká. 2003. The prague dependency treebank: A three-level annotation scenario. In A. Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*, pages 103–127. Kluwer.

V. Demberg and F. Keller. 2008. Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. In *Cognition*, volume 109, Issue 2, pages 193–210.

G. Gazdar, E. Klein, Pullum G., and I. Sag. 1985. *The Logic of Typed Feature Structures*. Blackwell.

F. Keller. 2004. The entropy rate principle as a predictor of processing effort: An evaluation against eye-tracking data. *Proceedings of the conference on empirical methods in natural language processing*, 317(324):324.

A. Koller and S. Thater. 2010. Computing weakest readings. In *ACL '10: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

D. Lin. 1998. A dependency-based method for evaluating broad-coverage parsers. *Natural Language Engineering*, 4(2):97–114.

H. Telljohann, H. Hinrichs, and Kübler S. 2004. The tüba-d/z treebank - annotating german with a context-free backbone. In *4th International Conference on Language Resources and Evaluation*.

K. Tomanek, U. Hahn, S. Lohmann, and J. Ziegler. 2010. A cognitive cost model of annotations based on eye-tracking data. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1158–1167.

B. Webber. 2009. Genre distinctions for discourse in the penn treebank. In *47th Annual Meeting of the ACL*, pages 674–682.

T. van der Wouden, H. Hoekstra, M. Moortgat, B. Renmans, and I. Schuurman. 2002. Syntactic analysis in the spoken dutch corpus. In *3rd International Conference on Language Resources and Evaluation*, pages 768–773.

```
<category label="SENT" sample:index="0" sentence:index="0:16" node:index="0:16:0">
   <category label="NP" features="NP:SUJ" node:index="0:16:1">
      <category label="Det" features="Da-ms----" node:index="0:16:2" form="Le" lemma="le"/>
      <category label="Noun" features="Ncms--" node:index="0:16:3" form="texte" lemma="texte"/>
      <category label="AP" features="AP" node:index="0:16:4">
         <category label="Adj" features="Af-ms-" node:index="0:16:5" form="mme" lemma="mme"/>
         <indices grammaticality="0.5866" sat:ratio="1.0" completeness="0.1176" quality:index="1.0" precision="0.5847"/>
      </category>
      <indices grammaticality="0.5993" sat:ratio="1.0" completeness="0.1525" quality:index="1.0" precision="0.6011"/>
   </category>
   <category label="VP" features="VP" node:index="0:16:6">
      <category label="VN" features="VN" node:index="0:16:7">
         <category label="Aux" features="Vaii3s--" node:index="0:16:8" form="avait" lemma="avoir"/>
         <category label="Verb" features="Vmps-smaip--" node:index="0:16:9" form="reu" lemma="recevoir"/>
         <indices grammaticality="0.6201" sat:ratio="1.0" completeness="0.2105" quality:index="1.0" precision="0.6284"/>
      </category>
      <category label="NP" features="NP:OBJ" node:index="0:16:10">
         <category label="Det" features="Ds3msp---" node:index="0:16:11" form="leur" lemma="leur"/>
         <category label="Noun" features="Ncms--" node:index="0:16:12" form="accord" lemma="accord"/>
         <category label="AP" features="AP" node:index="0:16:13">
            <category label="Adj" features="Af-ms-" node:index="0:16:14" form="formel" lemma="formel"/>
            <indices grammaticality="0.5851" sat:ratio="1.0" completeness="0.1176" quality:index="1.0" precision="0.5847"/>
         </category>
         <indices grammaticality="0.5981" sat:ratio="1.0" completeness="0.1525" quality:index="1.0" precision="0.6011"/>
      </category>
      <indices grammaticality="0.5871" sat:ratio="1.0" completeness="0.1111" quality:index="1.0" precision="0.5816"/>
   </category>
   <category label="Pct" features="Wd" node:index="0:16:15" form="." lemma="."/>
   <indices grammaticality="0.6224" sat:ratio="1.0" completeness="0.2142" quality:index="1.0" precision="0.6302"/>
</category>
```

Figure 9: Example of tree enriched with grammaticality

Semantic Annotation of the French Treebank with Modular Graph Rewriting

Bruno Guillaume[1, 2] **Guy Perrier**[1, 3]

(1) LORIA - Campus Scientifique - BP 239 - 54506 Vandœuvre-lès-Nancy cedex
(2) INRIA Nancy Grand Est - 615, rue du Jardin Botanique - 54600 Villers-lès-Nancy
(3) Université de Lorraine - 34, cours Léopold - CS 25233 - 54502 Nancy cedex
bruno.guillaume@loria.fr, guy.perrier@loria.fr

Abstract

We propose to annotate the French Treebank with semantic dependencies in the framework of DMRS starting from an annotation with surface syntactic dependencies and using modular graph rewriting. This system has been experimented on the whole French Treebank with the prototype which implements the rewriting calculus.

1. Introduction

We propose to produce a semantic annotation of large corpora from an annotation with syntactic dependencies. For the semantic annotation, we choose the framework of DMRS (Dependency Minimal Recursion Semantics). DMRS was introduced by (Copestake, 2009) as a compact and easily readable equivalent to Robust Minimal Recursion Semantics (RMRS), which was defined by (Copestake, 2007). This underspecified semantic formalism was designed for large scale experiments without committing to fine-grained semantic choices. DMRS graphs contain the predicate-argument relations, the restriction of generalized quantifiers and the mode of combination between predicates. Predicate-argument relations are labelled arg_i, where i is an integer following a fixed order of obliqueness *suj*, *obj*, *ats*, *ato*, *a-obj*, *de-obj*. . . .

We have chosen DMRS because we aim at an annotation which is readable and minimal. We want to avoid any commitment to questionable linguistic choices. Moreover, the DMRS structures are based on dependencies like our initial syntactic structures, which makes the transformation easier.

Regarding the input corpora, there are very few French resources syntactically annotated; the largest one is the *French Treebank*. The French Treebank is a corpus of sentences extracted from the newspaper “Le Monde”, which are annotated with phrase structures (Abeillé et al., 2003). These annotations have been converted into syntactic dependencies (Candito et al., 2009) following a format described in the guide designed for this task[1]. In our work, we use this last corpus under the abbreviation FTB.

To compute the semantic structures, we use the framework of *graph rewriting* (Bonfante et al., 2011), which is motivated by the shape of the manipulated objects. The output DMRS structures are graphs of semantic dependencies. The input syntactic structures are often dependency trees but to compute semantics, we need to complete these trees, for instance with some syntactic arguments of infinitives or some antecedents of pronouns determined by the syntax. We obtain wholly general graphs, with cycles and vertices that have several antecedents.

Term rewriting and tree rewriting can benefit from a canonical definition, whereas no such definition exists for graph rewriting. For our application, we have chosen an operational view of graph rewriting. A graph rewriting system is defined as a set of *rewrite rules*. Given a rule, modification of the graph (which correspond to the part usually called “right-hand side” of the rule) is defined by means of a set of commands; the control of the way the rule is applied (usually called the “left hand-side”) uses pattern matching as is done in a traditional graph rewriting setting.

In our modeling of the syntax-semantics interface, every linguistic principle is translated into a few simple and readable rules. Since our ambition is to process large corpora, a complete system can have several hundred rules. The more rules, the more interaction between rules, and the consistency of the whole rule system becomes difficult to maintain. This impinges on our ambition of a large coverage for the grammar. To solve this problem, we propose to organize rules in *modules*.

A module is a set of rules that is linguistically consistent and represents a particular step of the transformation. There is no order between rules inside a module but we fix the order between modules according to linguistic criteria. Modules play a decisive role in the construction of a rewriting system but they are also crucial for the efficiency of computations. The transformation of syntax into semantics is non confluent by essence because the correspondence between syntactic and semantic structures is relational and not functional. With the use of modules, it is possible to restrict non confluence to some particular modules.

Another original feature of our calculus is the way that it makes the link between rules and lexicons. During the syntax-to-semantics transformation, many rules have to be controlled by lexical information; these rules are called *lexical rules*. Of course, many rules differ only by lexical information they contain; that’s why lexical rules can be parametrized. Each lexical rule is associated with a lexicon, which provides a list of possible values for the parameters. Every time such a rule is applied, it is verified that the values of input parameters given by the graph in which the rule matches are present in the associated lexicon.

In our application, we use lexical rules for the detection of syntactic subcategorization frames of verbs and the transformation of syntactic arguments into semantic arguments.

[1] http://alpage.inria.fr/statgram/frdep/fr_stat_dep_parsing.html

To build the lexicons associated with lexical rules, we appeal to an external resource: Dicovalence (Van den Eynde and Mertens, 2003). Dicovalence is a lexicon of French verbs with more than 8000 entries. Each entry corresponds to a particular meaning of a verb and it gives rich information about its subcategorization frame. All entries have been validated manually by linguists.
Our rewriting system is composed of 562 rules including 402 lexical rules and organized in 34 modules. In the rest of the article, it is called SynSem$_{\mathbf{FTB}}$. Our calculus is implemented in a prototype which was used to experiment SynSem$_{\mathbf{FTB}}$ on the whole FTB.
The plan of the article is the following: Section 2. presents the main features of our graph rewriting calculus; the system SynSem$_{\mathbf{FTB}}$ of rules used for transforming the syntactic annotation of the FTB into a semantic annotation is described in Section 3. and finally Section 4. gives the results of the experimentation with the FTB.

2. The main features of the calculus

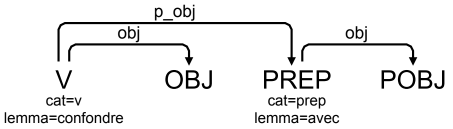
2.1. The rules

Each rule is composed of two parts: the *template* and the *commands*. To explain each one, we consider a rule R which is applied to a graph G:

- The template contains one *positive pattern* P and a possibly empty set of *negative patterns* $\{N_1, \ldots, N_k\}$. The positive pattern is a graph and each negative pattern N_i is a graph extension[2] of P. A rule can be applied if the positive pattern P match the graph G and, for all negative graph extension N_i, the matching of P into G fail to extend to a matching of $P \cup N_i$ in G.

- The commands part of the rule describes atomic operations that are used to modify the graph G when the rule R is applied. A command is defined relatively to the positive pattern of the rule; it can be addition or deletion of a node, an edge or a feature, merging of two nodes .... Commands are executed in order and then this order is relevant.

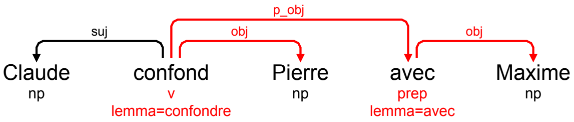
To illustrate our purpose, we consider the rule that transforms the syntactic arguments into semantic argument for the transitive verb *confondre* with an indirect object introduced with the specific preposition *avec* (*"confondre X avec Y"* translates to *"mistake X for Y"*). We call it `suj_V_obj_pobj`. Its positive pattern is represented with the following diagram:



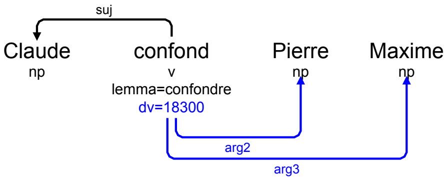
The complete rule is encoded in the following way:

```
1  rule suj_V_obj_pobj {
2    match{
3      V [cat=v, lemma="confondre"];
4      OBJ [];
5      objrel: V -[obj]-> OBJ;
6      PREP [cat=prep, lemma="avec"];
7      preprel:V -[p_obj]-> PREP;
8      POBJ [];
9      pobjrel:PREP -[obj]-> POBJ;
10   }
11   without{ V[dv=*] }
12   without{ V-[a_obj|de_obj|ato|ats]->* }
13   commands {
14     del_edge objrel; del_edge preprel;
15     del_edge pobjrel; del_node PREP;
16     add_edge V -[arg2]-> OBJ; add_edge V -[arg3]-> POBJ;
17     V.dv  = 18300;
18   }
19 }
```

Lines 2 to 10 describe the positive pattern P of `suj_V_obj_pobj` rule. Lines 11 and 12 describes two negative patterns N_1 and N_2. The first one (N_1 on line 11) prevents the rule to loop indefinitely: if the node V has already a feature named `dv`, the rule application is blocked. The second one (N_2 on line 12) prevents the matched verb to have other complements in the graph than a direct object (OBJ here) and an indirect object (POBJ here). Lines 13 to 18 describe the commands of the rule, which replace the direct object and indirect object dependencies with semantic dependencies, removing the preposition. Line 17 is used to record information about the lexical information used: in our case, we use Dicovalence which gives an integer identifier to each entry; it can be useful to keep track of this information.
Rule `suj_V_obj_pobj` applies to the following graph representing the syntax of the sentence *Claude confond Pierre avec Maxime (Claude mistakes Pierre for Maxime).*



The positive pattern of the rule matches with the part of the graph in red and the two negative patterns fail to extend this matching. Therefore, the rule applies and transforms the graph into the following one:



If some rule application on G produces G', we write $G \to G'$. If G rewrites to G' with any number of rule applications, we write $G \to^* G'$.

2.2. Lexical rules

The rule presented above uses lexical information. If we want to create a similar rule for every subcategorization frame of every verb, the best is to start from an existing lexicon of verbs. In our application to the FTB, we have chosen Dicovalence. Of course, we want to factorize the rules for different verbs which share the same subcategorization frame. Consider again the case of transitive verbs with an indirect object introduced with a specific preposi-

[2] A negative pattern can contain edges referring to nodes defined in P; formally, in negative conditions, we consider the graph defined by the union of $P \cup N_i$.

tion. The rule `suj_V_obj_pobj` defined above can be adapted to all transitive verbs with an indirect object introduced with the preposition *"avec"*. We can go one step further and consider that the preposition is also a parameter of the rule.

Thus, in order to avoid the increase in the number of rules, we define *parametrized rules*. In addition to the parametrization of the pattern, we also consider parameters in commands. Parametrized rules allow:

- several parameters (prefixed with the symbol `$` in rule code) to be used in the positive and negative patterns as input parameters;
- other parameters (prefixed with the symbol `@` in code) to be used in the commands as output parameters.

For instance, `suj_V_obj_pobj` rule is generalized into the parametrized rule below by changing its head (lines 2 and 3 in blue) and the value of some features (lines 6, 9 and 20 in red):

```
1  lex_rule suj_V_obj_pobj
2          (feature $lemma, $prep, @dicoval_id;
3           file "suj_V_obj_pobj.lp")
4  {
5  match{
6     V [cat=v, lemma=$lemma];
7     OBJ [];
8     obj: V -[obj]-> OBJ;
9     PREP [cat=prep, lemma=$prep];
10    pobj1:V -[p_obj]-> PREP;
11    POBJ [];
12    pobj2:PREP -[obj]-> POBJ;
13  }
14  without { V [frame=*] }
15  without {V -[a_obj|de_obj|ato|ats]-> *}
16  commands {
17    del_edge obj; del_edge pobj1;
18    del_edge pobj2; del_node PREP;
19    add_edge V -[arg2]-> OBJ; add_edge V -[arg3]-> POBJ;
20    V=@dicoval_id;
21  }
22 }
```

This rule refers, on line 3, to an external file (`suj_V_obj_pobj.lp`) which contains 151 entries coming from Dicovalence and corresponding to the concerned subcategorization frame. Each line of the file describes an entry: th<e lemma, the governed preposition and the identifier in Dicovalence:

```
accommoder#avec##900
accorder#avec##1090
accoupler#avec##1305
        ...
confondre#avec##18300
        ...
troquer#contre##84610
voir#en##86390
```

The two first elements are the possible values of the input parameters `$lemma` and `$prep` and the last one is the value of corresponding output parameter `@dicoval_id`.

If the new `suj_V_obj_pobj` rule is applied to the previous graph representing the syntax of the sentence *"Claude confond Pierre avec Maxime"*, the positive pattern matches in the same way but the matching entails the instanciation of the input parameters: `$lemma` with *"confondre"* and `$prep` with *"avec"*.

Then, ones verifies that the `suj_V_obj_pobj.lp` file contains the pair (*"confondre"*,*"avec"*). If it is the case, the output parameter `@dicoval_id` is instantiated with the corresponding value in the lexicon[3].

All lexical informations extracted form Dicovalence are automatically transformed into a set of rules (379 currently), each rule being associated with a file providing the possible instanciations of its parameters. When no directly usable resource exists, beginnings of lexicons have been built manually. This is the case for the subcategorization frames of adjectives and predicative nouns. This is also the case for scopal adverbs and non intersective adjectives.

2.3. Modules

The interest of rewrite rules is that their effect is local but it is a drawback if one considers the whole system of rules. It can contain several hundred of rules and so, if all rules are allowed to act in parallel, it is difficult to control their interaction. That's why the grouping of rules in modules and the ordering of modules is crucial.

Modularity makes the design, the maintenance and the adaptation of a rule system easier. A module is a set of rules that is linguistically consistent and represents a particular step of the transformation. For instance, in our proposal, there is a module transforming the syntactic arguments of verbs into their semantic arguments. Another module determines the semantic representation of attributive constructions.

Formally, a module is a finite set of rules. A module has the *termination property*, if for any graph G, there is no infinite rewriting $G \rightarrow G_1 \rightarrow G_2 \rightarrow \ldots$ starting from G. In this work; all the modules we consider have the termination property. Given a module M, a graph G is said to be a M-normal form if there is no rule in M that can be applied to G. For a module with the termination property, for any graph G, there is a finite number of M-normal forms G' such that $G \rightarrow^* G'$. A module is *confluent* if, when $G \rightarrow^* G_1$ and $G \rightarrow^* G_2$, there is a graph G' such that $G_1 \rightarrow^* G'$ and $G_1 \rightarrow^* G'$. The most important consequence of the confluence property is that it implies that every graph has an unique normal form and hence only one reduction path has to be computed; giving a much more efficient implementation.

In our rewriting system, rules are organized in a totally ordered list of modules that are applied in turn. Each module is applied to the list of normal forms of the previous modules. In the system presented in this article, there are 562 rules distributed between 34 modules. The global process we implement is not confluent; nevertheless it is possible to restrict the non-confluence to a small number of module: 26 of the 34 modules are confluent in our experiment.

2.4. Filters

Depending of the design of the module, normal forms for the rewriting calculus may not be linguistically consistent structures. For instance, consider the module that transforms the syntactic arguments of verbs into semantic argu-

[3] To guarantee that the application of the rule is deterministic, we assume that the relation between input and output parameters in the lexicon file is functional.

ments. After application of this module, some graphs may still include syntactic dependencies. It can happen that an intransitive verb remains with a direct object. Several reasons can explain this situation: a wrong annotation of the initial graph, a bad choice among the different uses of a verb, the non detection of an impersonal construction . . .
To remove inconsistent normal forms, we have introduced *filters*. A filter contains only a template (*i.e.* a positive pattern and some negative ones), it has no command part. Filters are defined inside a module and they are used in a post-processing step: among normal forms obtained with usual rules of the modules, normals forms that contains a filter template are removed. For instance, in the module transforming the syntactic arguments of verbs into semantic arguments, we have introduced a filter removing all graph in which some syntactic argument remains.

2.5. Rewriting complexity

Every rule has to perform the matching of a graph with a sub-graph of another graph. This problem in its whole generality is known to be NP-complete. Nevertheless, graph rewriting is used here in particular conditions:

- rule patterns are very small;
- all patterns of a rule are connected;
- the number of edges going out of a given pattern vertex is bounded (in our application, the bound is 4).

In these conditions, the matching of a rule with a graph is linear in time with respect to the size of the graph.
Moreover, to describe the syntax-semantics interface of natural languages, a general mechanism of vertex creation is not necessary. In fact, all vertices of the final structure are predictable from the initial structure. It is expressed in the rule definition in the following way: every vertex that is created must be attached to a vertex present at the entrance inside the module and it is possible to attach only a finite number of new vertices to a given vertex.
In our system, it is possible to define a measure on graphs which decreases proportionally to the square of the graph size. Therefore in all confluent modules, every computation runs in a time that is cubic with respect to the size of the input graphs.
To summarize, the computation complexity is not due to the chosen method, graph rewriting, but to the essence of the problem to which it applies: the production of a semantic representation from the syntax of a sentence. More precisely, the main source of complexity is lexical ambiguity.

3. The rewrite rule system

3.1. The organization of modules

The current SynSem$_{\mathbf{FTB}}$ system is composed of 562 rules including 402 lexical rules and organized in 34 modules. Figure 1 presents the system of modules as a graph, in which vertices represent modules and edges represent the order in the execution of modules.
These modules are themselves grouped in 6 packages:

- INIT transforms the initial CONLL annotation in a format compatible with the rules computing deep syntax;
- VERB is dedicated to the normalization of the verbal kernel: auxiliaries are deleted, transitive and pronominal verbs are marked, their voice is recognized as the active, passive or middle voice;
- SUJ is dedicated to the treatment of verb and adjective subjects[4];
- ARGV computes the deep syntactic arguments of verbs by redistribution of surface arguments (transformation of impersonal, causative, passive and middle constructions) and by lexical determination of some infinitive arguments;
- PRO normalizes the syntax of several pronouns and determines the antecedent of relative and reflexive pronouns;
- SEM computes the semantic relations between predicates and individuals coming from the deep syntactic dependencies between word lemmas.

The four packages VERB, SUJ, ARGV and PRO contribute to the production of the sentence annotation with deep syntactic dependencies. Hence they are gathered in the same super-package DSYNT. Then, the rules of the package SEM compute a semantic representation of sentences in the DMRS format. Consider the following sentence:

(1) *Jean a pu être autorisé à partir et à*
John might be allowed to leave and to
regagner son domicile.
return home.
John has managed to be allowed to leave and to return home.

Figure 2 illustrates the two step computations with sentence (1):

- the initial annotation with surface syntactic dependencies according to the annotation guide of the FTB,
- the annotation with deep syntactic dependencies resulting from the application of the rules from INIT, VERB, SUJ, ARGV and PRO packages,
- the semantic representation in the DMRS format after execution of the rules from SEM package.

Even if the number of rules in the system is high, the effect on the computation complexity is limited due to the shape of rules. For every rule, the patterns are connected graphs. These graphs are generally trees with a depth of 3, except for 5 rules from ANT_REL_PRO module, in which the patterns have two roots.
The main source of complexity comes from the non confluence of computations but non confluence is confined in 8 among the 34 modules[5].

[4] The systematic assignment of a subject to an adjective facilitates the subsequent treatment in a more uniform way.

[5] On Figure 1, they are represented with opaque ovals.

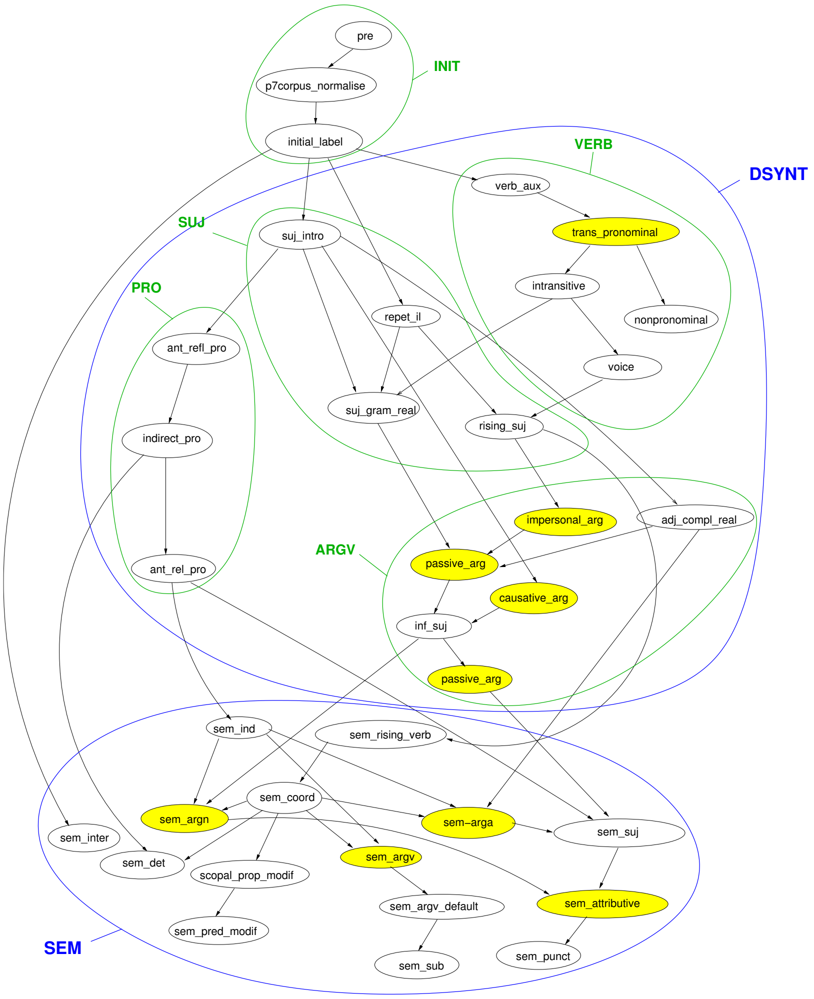
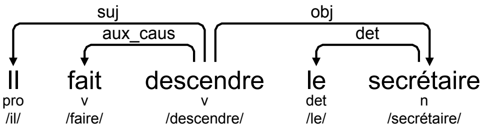


Figure 1: Diagram of the module system

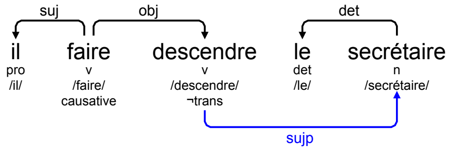
3.2. The grammatical coverage

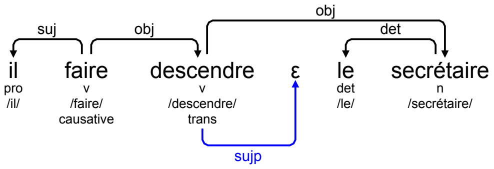
The SYNSEM$_{\mathbf{FTB}}$ system covers the most common grammatical phenomena in French related to the syntax-semantics interface, except comparative constructions and parenthetical expressions, which are very frequent in the corpus. Regarding lexical informations, Dicovalence gives a satisfying coverage for verbs but for the predicate nouns and adjectives, we have written small lexicons.

As an illustration, here are some significant examples. Consider causative constructions. In the FTB, a causative verb is viewed as an auxiliary of the complement infinitive. This infinitive is then the head of the verbal kernel and the governor of all its arguments. Here is the annotation in dependencies according to the FTB guide of the sentence *"il fait descendre le secrétaire"*:



As (Abeillé et al., 1997) shows it, this view is too simplistic and does not take some aspects into account. In the example above, *"secrétaire"* is the direct object of *"descendre"* but it is ambiguous: if *"secrétaire"* is a person [*"secretary"*], it can be the deep subject of *"descendre"* [*"to go down"*]; if it is furniture [*"secretaire"*] or if *"descendre"* is used with the meaning of [*"to kill"*], *"secrétaire"* is then the deep object of *"descendre"*. In order to distinguish the two readings, SYNSEM$_{\mathbf{FTB}}$ transforms the causative auxiliary into a full verb, which makes possible to express the two readings through the annotations below:





In both diagrams above, a dependency representing the deep subject of *"descendre"* has been introduced. In the first one, this subject is *"secrétaire"* and in the second one, it is not expressed in the sentence, which is represented with an empty word denoted ϵ. In a general way, all surface syntactic dependencies are represented over the text and the

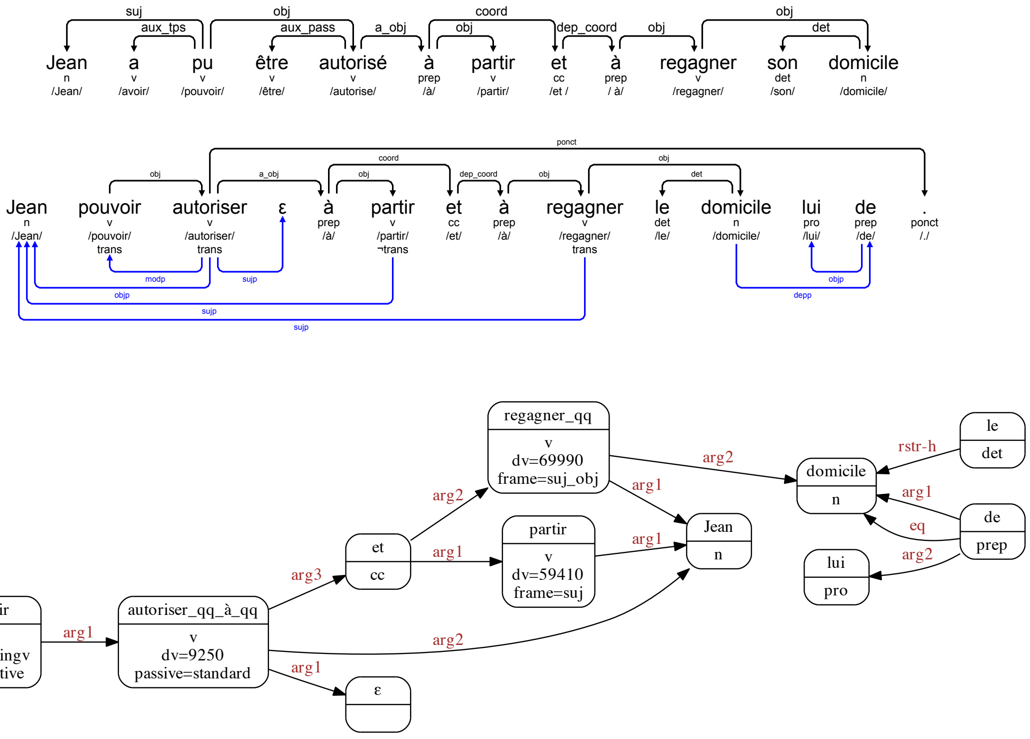


Figure 2: Surface syntax, deep syntax and semantics for the sentence *"Jean a pu être autorisé à partir et à regagner son domicile"*.

deep (syntactic and semantic) dependencies are drawn under the text. Deep syntactic dependencies have a label ending with "p" (*sujp*, *objp*, . . .) to be distinguished from surface syntactic dependencies.

Rising verbs are dealt with in the FTB like full verbs in the same way as control verbs. However there are good reasons (Rooryck, 1989) to consider them as hybrid objects which sometimes behave as auxiliaries. Consider the following examples where rising verbs are written in bold and their complement infinitives are underlined:

(2) *Il* **peut** *arriver* *deux* *personnes* *aujourd'hui.*
It may arrive two persons today.
'Two persons may arrive today.'

(3) *La* *maison* **peut** *être vendue* *aujourd'hui.*
The house may be sold today.

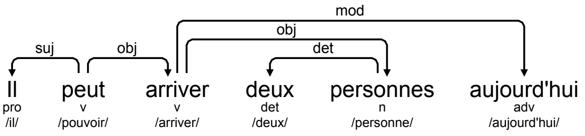
(4) *La* *maison* **peut** *se* *vendre* *aujourd'hui.*
The house may itself sold today.
'The house may be sold today.'

They successively illustrate an impersonal construction, a passive and a middle voice. To avoid the increase of the number of rules computing the semantic arguments, it is necessary to transform every construction into a canonical construction. Usually the canonical construction is the personal construction in the active voice. For the three previous examples, we obtain:

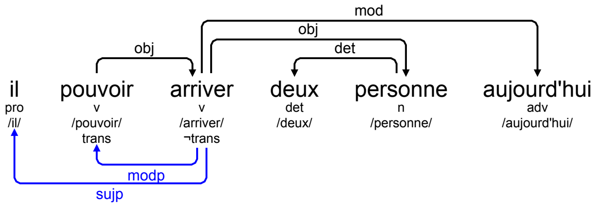
(2') *Deux* *personnes* **peuvent** *arriver* *aujourd'hui.*
Two persons may arrive today.

(3'), (4') *On* **peut** *vendre* *la* *maison* *aujourd'hui.*
One may sell the house today.

The rewrite rules used for this transformation are made simpler if the rising verbs are dealt with as auxiliaries. Then, the same rules as for verbs not depending on rising verbs can apply. Let us describe the method on example (2). Here is the initial annotation given by the FTB guide.

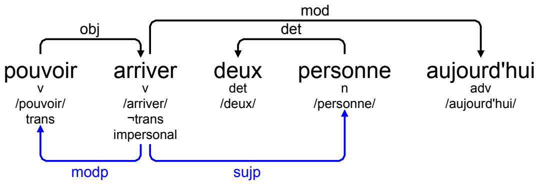


A first rule transforms the rising verb *"pouvoir"* into an auxiliary of *"arriver"* which becomes the sentence head.


mod
obj
obj
det
il pro /il/
pouvoir v /pouvoir/ trans
arriver v /arriver/ ¬trans
deux det /deux/
personne n /personne/
aujourd'hui adv /aujourd'hui/
modp
sujp

As the dependency structure is not restricted to be a tree, more expressive power is available. So the verb *"pouvoir"* keeps *"arriver"* as its object while being a modifier of it, which produces a cycle in the graph. As a consequence, the sentence head is no longer a tree root but it remains the distinguished vertex which is aimed at receiving any external dependency, especially when the sentence is inserted as a subordinated clause into another complex sentence.

After the transformation, the impersonal construction can be processed as any impersonal construction because it is anchored to the verb *"arriver"*. A rule aims at recovering the corresponding canonical construction. It removes the impersonal subject and the surface object becomes the deep subject of the verb. Hence, the resulting annotation:



3.3. Module Ordering

The organization of rewrite rules in modules is inseparable from the definition of a partial execution order for the modules. This order is determined by linguistic and representation choices. For instance, the module dedicated to impersonal constructions precedes the module dedicated to the passive voice so that impersonal passive constructions can be processed correctly.

Some cases may be very complicated. Consider the following examples:

(5) *Jean est* **autorisé** *à arriver plus tard.*
John is allowed to arrive later.

(6) *Jean* **demande** *à Marie d'être accompagnée*
John asks to Mary to be accompanied
par sa fille.
by her daughter.

In every sentence above, a verb in bold controls the subject of an infinitive which is underlined. For sentence (5), the module redistributing the passive construction, denoted PASSIVE_ARG, must apply first to produce the following canonical construction:

(5') *On* **autorise** *Jean à arriver plus tard.*
One allows John to arrive later.

Then, the module determining the subject of infinitives depending on control verbs, denoted INF_SUJ, can apply. In this module, a lexical rule indicates that the subject of *"arriver"* is the direct object of *"autorise"*.

For sentence (6), it is the contrary. Module INF_SUJ must be applied first to find that the subject of *"être accompagnée"* is the indirect object of *"demande"*. Then, module PASSIVE_ARG is applied to transform the passive voice into the active voice:

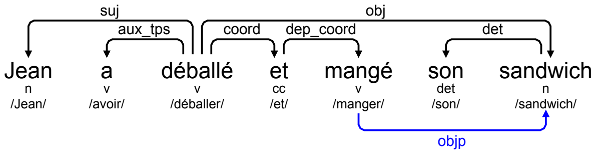
(5') *Jean* **demande** *à Marie que sa fille*
John asks to Mary that her daughter
l' accompagne.
her accompanies.
John asks to Mary that her daughter accompanies her.

Hence, in the order of module application, is present the sequence PASSIVE_ARG, INF_SUJ and again PASSIVE_ARG, as diagram 1 shows it.

Coordination is very specific and cannot be dealt with as most syntactic constructions within a particular module, which would be inserted somewhere in the module ordering graph. It allows sharing between structures and thus interacts with other syntactic constructions. This interaction is modeled with rules which are distributed between the different modules computing the deep syntax. Of course, these rules depend on the choice made for annotating coordination in the FTB; in particular, imposing the dependency structure to be a tree has strong consequences. So, the sharing of structures frequently introduced by coordination cannot be expressed because it needs that two dependents share the same governor. Rules of SynSem$_{\mathbf{FTB}}$ are dedicated to recovering sharing between coordinated structures. Depending on whether the shared structure is the subject of a verb, a passive or tense auxiliary or the antecedent of a relative pronoun, the phenomenon is respectively dealt with in the module introducing subjects (SUJ_INTRO), removing auxiliaries (VERB_AUX), or finally determining the antecedent of relative pronouns (ANT_REL_PRO).

The sharing of complements is more difficult to deal with because some complements may be optional, which makes the annotation of the FTB ambiguous. Consider the following sentence annotated according to the FTB guide.

(7) *Jean a déballé et mangé son sandwich.*
John has unpacked and eaten his sandwich.



The *objp* dependency under the sentence must be added to express that *"sandwich"* is shared by *"déballé"* and *"mangé"*. Unfortunately, the FTB does not allow such a dependency because *"sandwich"* would have two governors, which violates the treeness constraint on the dependency structure. Therefore, it is not possible to distinguish the sentence in which *"son sandwich"* is shared, from sentence *"Jean a déballé son sandwich et mangé"* if one merely takes the dependency structure into account. Only the linear order between words allows the distinction. In SynSem$_{\mathbf{FTB}}$ we have chosen to leave this problem aside provisionally.

4. Experimental results

The SynSem$_{\mathbf{FTB}}$ module system has been applied to the 12 351 sentences of the FTBwith the Grew[6] software. Complete statistics about the rule use by module on the whole corpus and detailed results (with the produced structures) on 1% of the corpus are available online[7].

Since computations are not confluent, we are interested first in the number of normal forms produced for each input sen-

[6] http://grew.loria.fr

[7] http://wikilligramme.loria.fr/doku.php?id=lrec2012

aff	arg	dep	aux_caus	aux_tps	aux_pass	comp	coord	dep_coord	det
1581	465	32821	128	3806	1672	37	6358	7299	40355
0	366	2240	5	26	15	37	194	1205	597
0,0%	78,7%	6,8%	3,9%	0,7%	0,9%	100,0%	3,1%	16,5%	1,5%

mod	mod_rel	ponct	suj	obj	a_obj	de_obj	p_obj	ato	ats
58353	2352	35810	15100	58132	2192	1668	1663	160	2521
13914	407	10598	158	3127	15	23	4	15	15
23,8%	17,3%	29,6%	1,0%	5,4%	0,7 %	1,4%	0,2%	9,4%	0,6%

Table 1: Dependency types processed by the SynSem$_{\mathbf{FTB}}$ rewriting system

tence. On one hand, long sentences can be highly ambiguous. On the other hand, filters at the end of some modules can discard an important number of inconsistent annotations. Some sentences may even have no normal form; it is the case for 16.0% of the sentences. For 33.1% of the sentences, the system returns one normal form, for 20.6%, it returns two normal forms. 92.3% of the sentences lead to a number of normal forms that is less than or equal to 8.

To estimate the coverage of the SynSem$_{\mathbf{FTB}}$ system, we have observed the number of dependency types that are present in the FTB and that are ignored in the rewriting process. Table 1 shows the number of dependencies present in the resulting annotation with respect to the number present in the initial annotation for every dependency type and for the sentences having one normal form at least.

It is difficult to distinguish the cases coming from an annotation error from the cases that are not covered by SynSem$_{\mathbf{FTB}}$. Nevertheless, for dependency types having more than 5% of occurrences remaining at the end, we can give some comments. Since comparatives are not dealt with by SynSem$_{\mathbf{FTB}}$, the *comp* dependencies are not rewritten. Punctuation dependencies related to parenthetical expressions are not dealt with too. The remaining *mod* dependencies mainly concern nouns modifying nouns or verbs and the remaining *mod_rel* often correspond to relative clauses in which the head verb is missing. For coordination, it is not rare to find several *dep_coord* dependencies starting from the same governor, which is an annotation error. *dep* dependency type is underspecified and used in various contexts, which are not all predicted by the FTB annotation guide.

Finally, we are interested in the use of Dicovalence in the rewriting of the FTB. The FTB contains 39 104 verbs and the sentences having at least one normal form contain 29 782 verbs. For these sentences, we find 17 542 verbs associated with a Dicovalence entry in the final annotation. It represents 58.9% (17 542 out of 29 782) of the rewritten verbs and 44,9% (17 542 out of 39 104) of all verbs.

We have studied the inconsistency cases in a precise way on the 1% of the corpus that is available online. It is composed of 120 sentences and 12 of them lead to no normal form. For 5 of them, Dicovalence does not describe the needed subcategorization frame: *"hispaniser"* transitive,*"acheter"* with indirect object and without direct object,*"maintenir"* with locative complement,*"souscrire"* in its transitive use and *"vendre"* in its intransitive use. Other cases correspond to annotation errors.

Conclusion

Through the example of the FTB, we have shown that graph rewriting is relevant for the automatic annotation of large corpora with semantic dependencies starting from surface syntactic dependencies.

Modules play a major role in the rewriting system both for controlling computations and maintaining the global consistency of the system. SynSem$_{\mathbf{FTB}}$ has been designed according to particular input and output formats, but the modularity of the system allows its adaptation to other formats at minimal cost. The fact that the module PASSIVE_ARG is used twice in the list of modules is not satisfactory. We can imagine sentences where it must be used more and it suggests to enrich the strategy language with the possibility to apply iteratively a sequence of modules until a fixpoint is reached; we leave this for further work.

Regarding the specific application to the FTB, the resulting annotation presents two main limits: the initial annotation contains many errors and the formal framework chosen for semantic annotation, DMRS, offers no solution for the modeling of some semantic properties (intentionality, comparatives, . . .). We hope to push the first limit by using graph rewriting for correcting the most systematic errors. The solution to the second limit depends on advances in formal semantics.

5. References

A. Abeillé, D. Godard, and P. Miller. 1997. Les causatives en français, un cas de compétition syntaxique. *Langue française*, 115:62–74.

A. Abeillé, L. Clément, and F. Toussenel, 2003. *Building a Treebank for French*, chapter 10. Kluwer Academic Publishers.

G. Bonfante, B. Guillaume, M. Morey, and G. Perrier. 2011. Modular graph rewriting to compute semantics. In *IWCS 2011*, pages 65–74, Oxford, UK, January.

M.-H. Candito, B. Crabbé, P. Denis, and F. Guérin. 2009. Analyse syntaxique statistique du français : des constituants aux dépendances. In *Actes de TALN 2009 (Traitement automatique des langues naturelles)*, Senlis, June. ATALA, LIPN.

A. Copestake. 2007. Semantic composition with (robust) minimal recursion semantics. In *Proceedings of the Workshop on Deep Linguistic Processing*, pages 73–80. Association for Computational Linguistics.

A. Copestake. 2009. *Invited Talk:* Slacker semantics: Why superficiality, dependency and avoidance of commitment can be the right way to go. In *Proceedings of EACL 2009*, pages 1–9, Athens, Greece.

J. Rooryck. 1989. Les verbes à montée et à contrôle "ambigus". *Revue québécoise de linguistique*, 18(1):189–206.

K. Van den Eynde and P. Mertens. 2003. La valence : l'approche pronominale et son application au lexique verbal. *French Language Studies*, 13:63–104.

An Open Infrastructure for Advanced Treebanking

Victoria Rosén*§, Koenraad De Smedt*, Paul Meurer§, Helge Dyvik*§

University of Bergen* and Uni Research§
Bergen, Norway
victoria@uib.no, desmedt@uib.no, paul.meurer@uni.no, dyvik@uib.no

Abstract

Increases in the number and size of treebanks, and the complexity of their annotation, present challenges to their exploration by the research community. Adhering to different formalisms, lacking clear standards, and requiring specialized search and visualization and other services, treebanks have not been widely accessible to a broad audience and have remained underexploited. The INESS project is providing the first infrastructure integrating treebank annotation, analysis and distribution, bringing together treebanks for many different languages, spanning different annotation schemes and including parallel treebanks. The infrastructure offers a uniform interface, interactive visualizations, leading edge search capabilities and high performance computing.

1. Introduction

Treebanks have without any doubt become one of the most powerful kinds of language resources. Parsers with probabilistic components trained on treebanks are now regarded as indispensable for wide coverage analysis and are therefore a prerequisite for realistic applications such as high quality syntax-based machine translation. Moreover, treebanks have a potentially wide user group including also general linguists and language scholars, but these groups still face obstacles in accessing the knowledge embedded in the resources.

The Penn Treebank (Marcus et al., 1993) has been influential as a standard resource and benchmark during the past couple of decades. Treebanks have been developed for a large number of languages, they have become larger in size, and their linguistic annotation is becoming richer, although treebanks with highly detailed syntactic and semantic analyses are still scarce. The field has some *de facto* standards for simple treebank formats but lacks comprehensive technical and organizational solutions for handling the variety of different formalisms, annotation standards and encodings.

Furthermore, the produced treebanking resources and tools are scattered on many sites, each with their own access policies and formats, and some lack curation and archiving policies. By way of example, we mention the German Tiger corpus and the TigerSearch tool which are potentially very useful but are no longer maintained by the creators.

An open infrastructure for the curation and dissemination of treebanks is therefore a timely goal. By ‘infrastructure’ we mean a persistent, integrated and managed set of services combining data and tools. By ‘open’ we mean that the system is not limited to a narrow set of data or users, that any researcher in principle can deposit, access and process data. In establishing such open infrastructures, we are moving from passive repositories towards online eScience laboratories which are easy to access and can address a variety of user needs.

Some usage scenarios of very large treebanks have been explored in the Dutch LASSY project, which has produced huge parsed corpora (van Noord, 2009). One such scenario relates to an investigation of conditions on extraposition, a construction where a constituent is discontinuous (cf. the English example *The question [is raised] why the government does not fund more research*). Since it is difficult to find hard rules governing the conditions under which extraposition can apply, an empirical investigation may be in order. However, such an investigation will hardly be possible in plain text corpora, not even in corpora tagged with parts of speech. Crucially, only a syntactically analyzed corpus provides the required level of detail that allows a systematic search with reliable results, as convincingly demonstrated by van Noord (2009). Enabling linguists and other users to address such questions in a user-friendly way across treebanks with different annotations and for different languages, but using a single access point, is a worthwhile goal.

Building a high quality treebank always requires a big investment, as human contributions in the form of linguistic insight and manual quality control are indispensable in addition to automatized procedures. It is therefore important to get a high return on investment by securing the usability of the finished treebank. More and more attention is being paid to archiving and disseminating treebanks, with appropriate documentation and licenses. Some approaches are illustrated by the Prague Dependency Treebank (Hajič, 1998; Böhmová et al., 2003) and the Icelandic Parsed Historical Corpus (IcePaHC) (Wallenberg et al., 2011). The former is distributed by the LDC and browsable on the web, has a bespoke license, and is searchable with an downloadable application (TrEd 2.0) as well as through a client-server application (Netgraph). IcePaHC is archived with versioning, has direct open download links, is released under a LGPL licence and is searchable with the downloadable CorpusSearch 2. Neither is fully accessible through web-based services from a browser.

2. INESS

The INESS project[1], running from 2010 to 2015, is probably the first large scale project aimed at building an eScience infrastructure for the exploration of syntax and semantics based on treebanking, with a wide range of resources and

[1] Infrastructure for the Exploration of Syntax and Semantics, http://iness.uib.no

services. This infrastructure has been operational on an experimental basis since 2010 and has been steadily expanded and adapted. It is not only the project's aim to make it easy for the R&D community to find, filter and download treebanks, but also to let the community actively participate in uploading and annotating treebanks. Furthermore, one of our goals is to provide a more uniform treatment of treebanks so that they can be linked and explored in similar ways.

The most general characteristic of the INESS infrastructure is that its services are fully accessible through any modern web browser, without the need to download and install any other software on the user's platform. The server middleware was written in Common Lisp on top of an open source web server in the same language.[2] The use of the same high level programming language throughout the whole system has resulted in a highly flexible system in which all annotation and analysis services are seamlessly integrated. The system is easy to modify at all levels, which promotes a fast evolution in response to user needs. Visualizations are based on Scalable Vector Graphics (SVG), which is supported by modern web browsers. The remainder of this article will present the status of the INESS infrastructure.

3. Selection of treebanks

While the INESS project is partly devoted to developing a large treebank for Norwegian, the infrastructure is open to hosting any other treebanks which may be useful in research. Currently the INESS middleware can handle LFG, constituency and dependency treebanks in various formats. INESS invites treebanking projects to deposit their treebanks in the infrastructure in order to make them accessible, and it currently provides access to 53 treebanks, ranging from small test suites to full size treebanks. This steadily growing number has made it necessary to provide a search interface at the metadata level. The user can make a choice of treebanks by selecting values for the following criteria:

- Language: All · Norwegian Bokmål (11) · German (6) · Georgian (5) · Hungarian (4) · Latin (4) · Church Slavic (3) · Ancient Greek (to 1453) (3) · Icelandic (2) · Northern Sami (2) · Wolof (2) · Classical Armenian (2) · Abkhazian (1) · Danish (1) · Estonian (1) · Gothic (1) · Norwegian Nynorsk (1) · Swedish (1) · Tigrinya (1) · Turkish (1) · Urdu (1)
- Collections: All · GeoGram (3) · HunGram (4) · IcePaHC (1) · NorGram (8) · PROIEL (13) · Sofie (8) · Test (5) · TiGer (3) · XPar (3)
- Annotation types: All · lfg (30) · dependency-proiel (13) · constituency (8) · dependency-cg (2)

For the languages, ISO-639-3 codes are internally used.

The annotation types currently distinguish between the following: *lfg* (Lexical Functional Grammar); *dependency-proiel*, a dependency annotation used in the PROIEL project,[3] based on dependency grammar enriched with secondary dependencies reminiscent of the structure sharing mechanism in LFG; *constituency*, which provides simple phrase structure constituency; and *dependency-cg*, based on Constraint Grammar.

Collections are loosely defined groups of treebanks based on similar texts or on similar grammars used in the analysis. For instance, *GeoGram* is a collection of materials parsed with the same Georgian grammar while *HunGram* is a collection parsed with the same Hungarian grammar. *Sofie* is a collection based on text from the novel *Sofies verden* [Sophie's World] (Gaarder, 1991) and its translations, but parsed with different grammars, an action initiated by the Nordic Treebank Network (Nivre et al., 2005). The parallel Sofie treebanks were collected, catalogued and aligned in cooperation with the META-NORD project.[4].

According to the user's choices, a list of treebanks is presented. The resulting list is the intersection between the values for the three criteria; however, each criterion allows multiple values of which the union is taken. For instance, choosing both the NorGram and Sofie collections selects all treebanks from both collections. This is illustrated in Figure 1 where these chosen NorGram and Sofie collections are in boldface. Furthermore, in this example Norwegian Bokmål is selected as the language, which means that only Norwegian treebanks are chosen from these collections. The chosen annotation type in this case was *All*. Because all treebanks chosen in this way are of type *lfg* or *constituency*, these are also automatically marked in boldface.

In the future, it will also be possible to select treebanks based on metadata attributes such as owner, licensing conditions, etc.

4. Visualization

Visualization is a nontrivial need for the exploration of highly detailed treebanks. Once a treebank is selected, its sentences are listed on the Sentence Overview page. Clicking on a sentence shows its structure by means of visualizations dependent on the annotation formalism as well as user preferences. For instance, the same German sentence from the Sofie constituency treebank can be visualized with a traditional tree structure as in Figure 2 or with Tiger-style horizontal and vertical lines as in Figure 3.

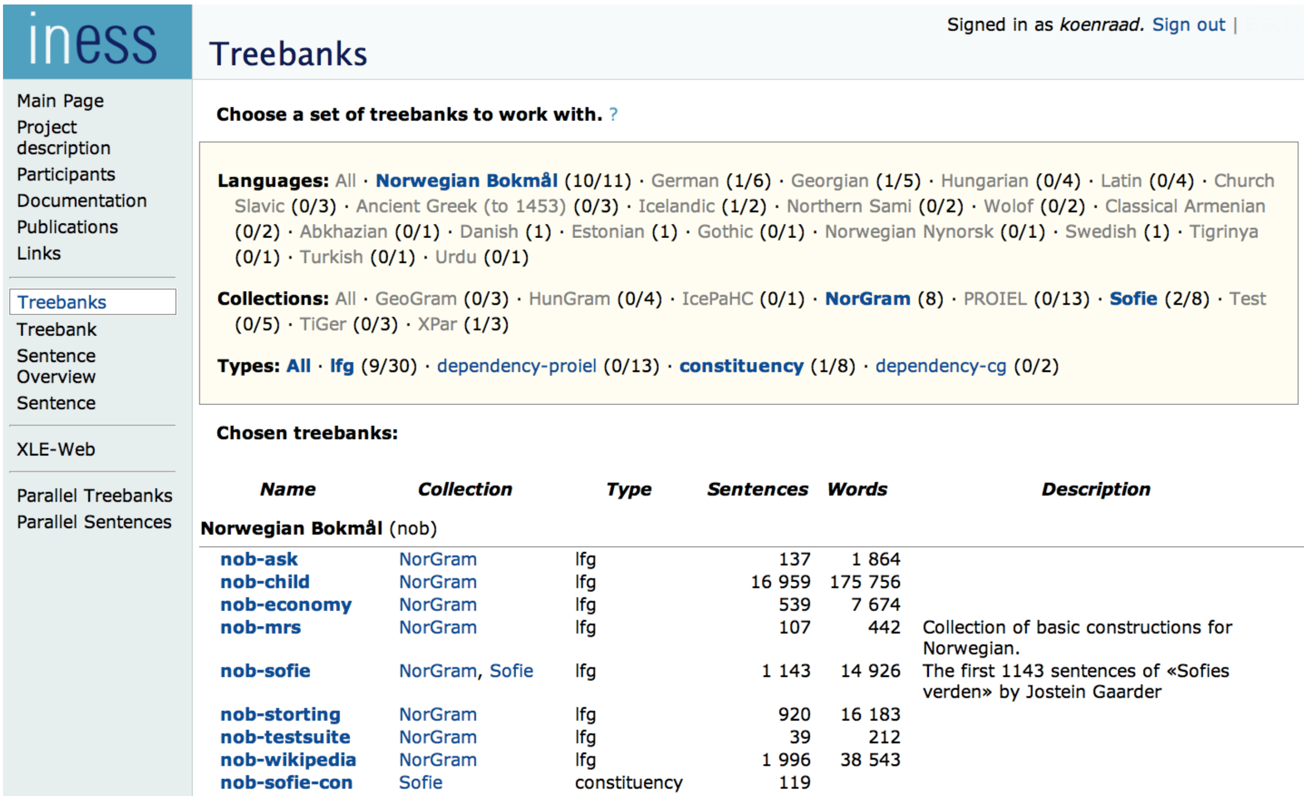
Structures in the LFG formalism are well supported through the integration of the LFG Parsebanker tool (Rosén et al., 2009), originally developed in the TREPIL project for the construction and exploration of LFG parsebanks. LFG treebanks are highly detailed and contain several levels of representation such as c-structures (constituent structures) and f-structures (functional structures, feature-value matrices). These structures are juxtaposed in the interface, with mouse-over highlighting to indicate corresponding elements in both structures. This is illustrated in Figure 4, where placing the cursor at PROPP in the c-structure causes highlighting of the value of the TOPIC feature in the (simplified) f-structure.

Parallel treebanks are visualized by displaying structures for aligned sentences in different languages next to each other. In Figure 5, German and Swedish constituency structures

[2] AllegoServe, http://allegroserve.sourceforge.net/.

[3] http://www.hf.uio.no/ifikk/english/research/projects/proiel//

[4] http://www.meta-nord.eu, under the umbrella of META-NET and linked to META-SHARE



Choose a set of treebanks to work with. ?

Languages: All · **Norwegian Bokmål** (10/11) · German (1/6) · Georgian (1/5) · Hungarian (0/4) · Latin (0/4) · Church Slavic (0/3) · Ancient Greek (to 1453) (0/3) · Icelandic (1/2) · Northern Sami (0/2) · Wolof (0/2) · Classical Armenian (0/2) · Abkhazian (0/1) · Danish (1) · Estonian (1) · Gothic (0/1) · Norwegian Nynorsk (0/1) · Swedish (1) · Tigrinya (0/1) · Turkish (0/1) · Urdu (0/1)

Collections: All · GeoGram (0/3) · HunGram (0/4) · IcePaHC (0/1) · **NorGram** (8) · PROIEL (0/13) · **Sofie** (2/8) · Test (0/5) · TiGer (0/3) · XPar (1/3)

Types: **All** · **lfg** (9/30) · dependency-proiel (0/13) · **constituency** (1/8) · dependency-cg (0/2)

Chosen treebanks:

Name	*Collection*	*Type*	*Sentences*	*Words*	*Description*
Norwegian Bokmål (nob)					
nob-ask	NorGram	lfg	137	1 864	
nob-child	NorGram	lfg	16 959	175 756	
nob-economy	NorGram	lfg	539	7 674	
nob-mrs	NorGram	lfg	107	442	Collection of basic constructions for Norwegian.
nob-sofie	NorGram, Sofie	lfg	1 143	14 926	The first 1143 sentences of «Sofies verden» by Jostein Gaarder
nob-storting	NorGram	lfg	920	16 183	
nob-testsuite	NorGram	lfg	39	212	
nob-wikipedia	NorGram	lfg	1 996	38 543	
nob-sofie-con	Sofie	constituency	119		

Figure 1: Screenshot of the treebanks selection interface

are juxtaposed. Phrase-aligned treebanks are also catered for, thanks to a methodology developed in the XPAR project (Dyvik et al., 2009). An example of phrase-aligned c- and f-structures in Georgian and Norwegian is given in Figure 6.

5. INESS-Search

Powerful tools for interactively searching and filtering treebanks are of primary importance in a treebanking infrastructure. The query syntax should be expressive in order to cater to a variety of research needs and the implementation should be efficient for a fast turnaround when searching a very large treebank. Furthermore, a search tool should be as simple and uniform as possible across different annotation types.

Currently a number of corpus search tools support searching and viewing of treebanks, such as CorpusSearch 2,[5] TrED 2.0,[6] and TIGERSearch.[7] They have a query language adapted to syntactic annotation in certain formats. TIGERSearch also includes a graphical query building interface. An overview of treebank query systems can be found elsewhere (Lai and Bird, 2004). Most search and viewing tools need to be downloaded and installed on the user's machine. TIGERSearch, which is no longer maintained, has been reimplemented as INESS-Search in Common Lisp, its functionality has been expanded, its query language has been made simpler and it has been integrated into the INESS web interface. INESS-Search can be used to query constituency and dependency treebanks, but it also contains extensions which are necessary for querying LFG f-structures, which are directed, possibly cyclic graphs rather than trees. An evaluation of INESS-Search against TIGERSearch and some other treebank search systems based on the TIGER treebank shows that the former is as fast or significantly faster on most types of queries (Meurer, 2012 forthcoming). Whereas the expressive power of TIGERSearch merely equals that of the existential fragment of first-order predicate logic over node variables (all node variables are implicitly existentially quantified), INESS-Search implements full first-order predicate logic. Its implementation in Common Lisp is seamlessly integrated in the infrastructure and it can therefore be used via a web interface in a straightforward way. This tight integration also means that search can be dynamic, i.e. changes in the treebank are immediately accessible to the search mechanism. This is particularly useful during the construction phase of the treebank when changes are frequent.

A graphical query construction tool has not been developed for INESS-Search since the query syntax is compact and intuitive enough (after some practice) to make such a device unnecessary. Moreover, it would be difficult to expose the full query syntax (including negation, disjunctions and quantifier scoping) in an elegant, easy-to-use graphical tool; implementing only the easier parts of the query syntax (like the existential fragment) would unneccessarily restrict the user to a less expressive subset of the language.

A mechanism for displaying and exporting search results in

[5] http://corpussearch.sourceforge.net/

[6] http://ufal.mff.cuni.cz/tred/

[7] http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch/oldindex.shtml (Lezius, 2002)

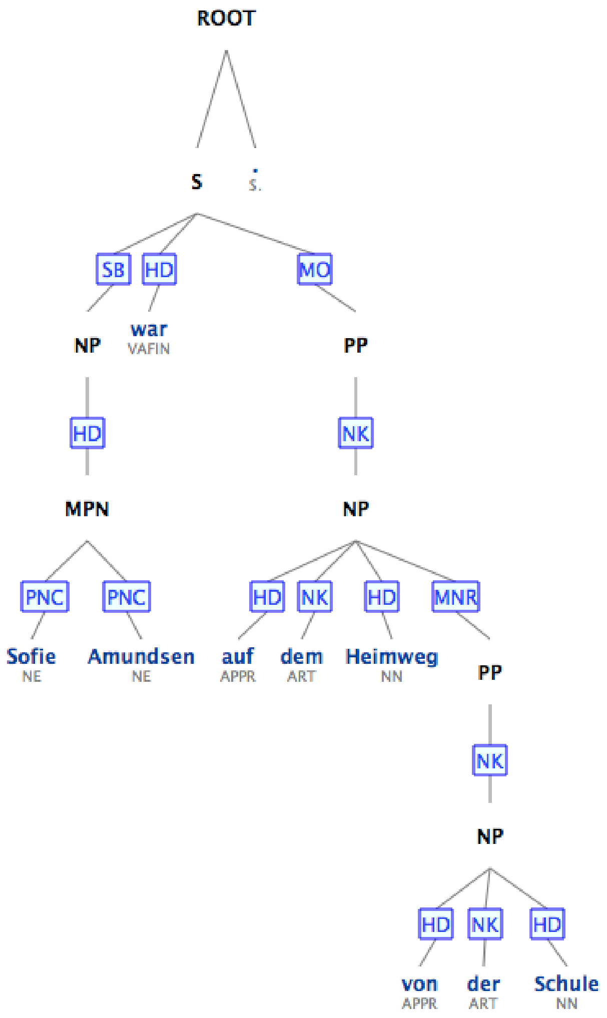


Figure 2: Screenshot of a constituent visualization, traditional branches

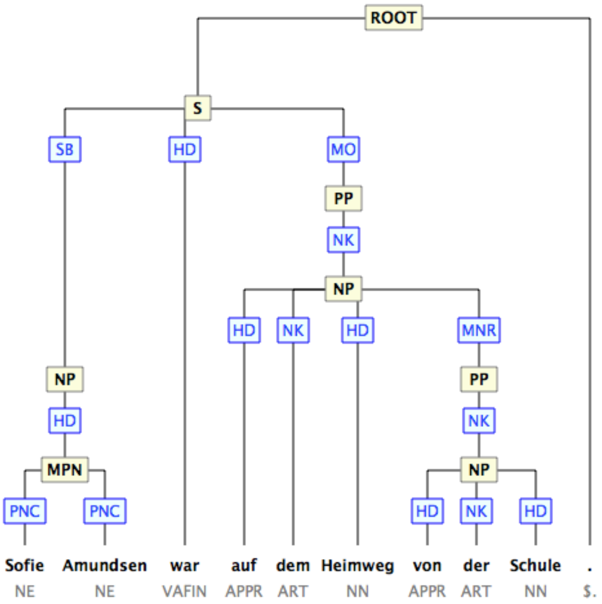


Figure 3: Screenshot of a constituent visualization, TIGER style branches

a flexible way is currently under development.

We may consider a few query examples. If one wants to find all sentences containing NPs that immediately dominate APs, the TIGERSearch expression in 1 can be used to that effect.

(1) [cat =“NP”] > [cat = “AP”]

In INESS-Search one may also use query 1, but the abbreviated syntax in 2 has the same results.

(2) NP > AP

If one wants to find all sentences containing NPs that immediately dominate APs that in turn immediately dominate PPs, variables are needed in TIGERSearch, as illustrated in 3.

(3) [cat = “NP”] > #x:[cat = “AP”] & #x > [cat = “PP”]

In INESS-Search the simplified expression in 4 has the same results as 3.

(4) NP > AP > PP

One result from this search in the Tiger treebank is shown in Figure 7, where the categories in the search expression are highlighted in red.

The query intentions in the examples in 5 are not expressible in TIGERSearch due to the lack of universal quantification.[8]

(5) Q2: Find sentences that do not include the word “saw”.
Q5: Find the first common ancestor of sequences of a noun phrase followed by a verb phrase.

The examples in 6 are INESS-Search queries expressing the intentions in the examples in 5.

(6) Q2: !(#x:“saw” = #x)
Q5: #c >∗ #n:NP !>∗ #v & #c >∗ #v:VP !>∗ #n
& !(#c >∗ #x >∗ #n & #x >∗ #v & #n .∗ #v)

As a convention, variables like c, v, and n in example 6 Q5 that occur in positive contexts are treated as existentially quantified, whereas variables like x in Q2 and Q5 that only occur in negated contexts are taken to be universally quantified and in the scope of all existential quantifiers.

A variable occuring in a positive context can be explicitly marked as universally quantified by using ‘%’ as the variable marker instead of ‘#’. In case the intended quantifier scoping deviates from the default, the scoping order can be given explicitly, as illustrated in query example 7 to search for all sentences where each NP dominates an N:

(7) (%x #y): %x:NP > #y:N

[8] Queries Q2 and Q5 are taken from Lai and Bird´s survey of treebank query systems (Lai and Bird, 2004), where they list typical queries that a query system should be able to express. Admittedly, the results of Q2 would be rather uninteresting to many.

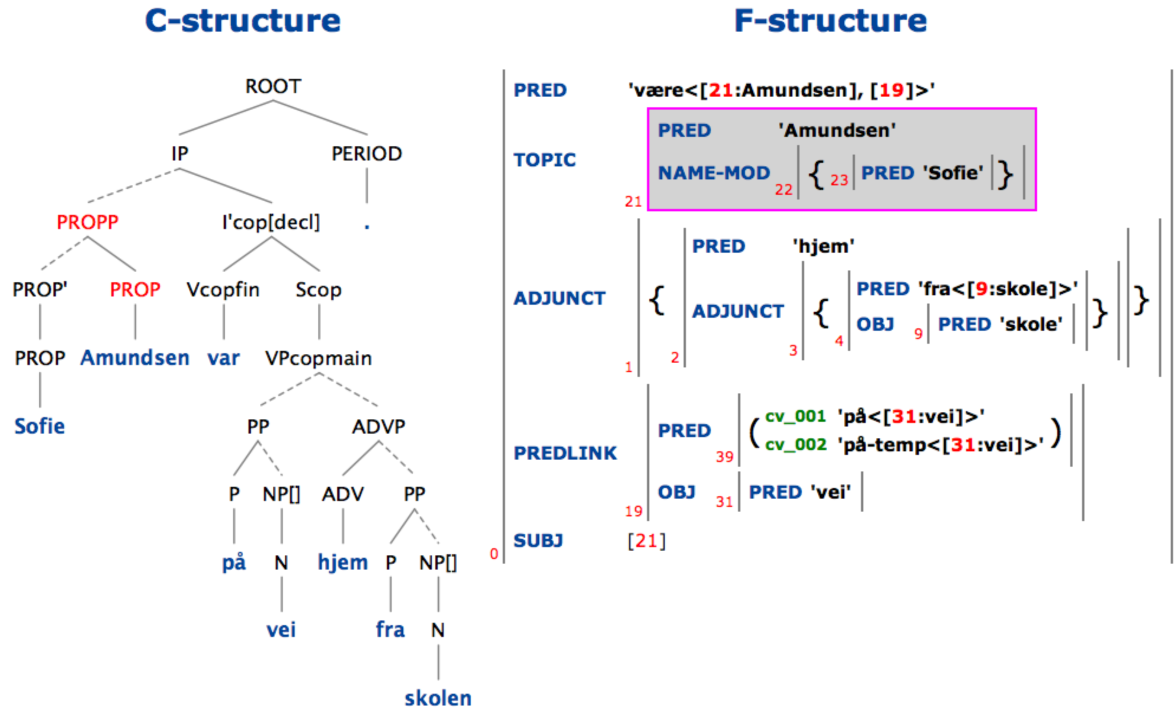


Figure 4: LFG c-structure and corresponding f-structure with mouse-over highlighting

Several operators have been implemented that allow for conveniently querying more complex tree node or f-structure constellations. A rule operator can be used for specifying parent–children constellations (e.g., 8). Operators specifically tailored to LFG structures are a projection operator, an extended-head operator, and regular expressions over f-structure attributes (e.g., 9).

(8) #c → AP .∗ PP

(9) #f >(TOPIC & XCOMP* OBJ) #g

In addition, functionality for querying parallel treebanks (Dyvik et al., 2009) is being further developed.

6. Interactive annotation

INESS offers advanced tools for the online interactive construction of treebanks, in particular LFG treebanks. The LFG Parsebanker tool (Rosén et al., 2009) was developed for this purpose. It was inspired by the [incr tsdb()] environment (Bender et al., 2011), a further development of the TSNLP methodology (Oepen and Flickinger, 1998), which supports annotation and grammar development through test suite management and regression testing. Our approach to treebanking has much in common with the approach advocated there, with the parsed corpus itself constituting part of the grammar development tool. But whereas their approach is mostly applied to HPSG grammars, ours is specialized for LFG grammars.
The LFG Parsebanker has been fully integrated into the infrastructure and offers the following workflow:

- A corpus is batch parsed with XLE (Maxwell and Kaplan, 1993; Kaplan et al., 2002) and all analyses (packed) are stored.
- For each sentence, discriminants are computed (Rosén et al., 2007) and presented to the annotator for disambiguation, as illustrated in Figure 8.
- The annotator's choice of discriminants is applied to the parse result and the remaining structures are displayed. This process can be repeated until the sentence is disambiguated.
- The chosen discriminants are stored; they can be undone by the annotator or automatically reapplied after reparsing.

Furthermore, a system for comments and issue tracking is provided to further assist in grammar development. Statistics are kept to measure discriminant frequencies and inter-annotator agreement. An integrated web-based parsing platform, the XLE-Web interface (Rosén et al., 2005), allows interactive parsing of sentences entered by the user. The infrastructure thus offers a complete online environment for the construction of LFG treebanks, without the need to download and install any software. This setup is currently being used to construct a number of LFG treebanks online for different languages, including a large Norwegian treebank.

7. Conclusion and future work

Treebanks are potentially highly useful, but high quality treebanks are very expensive to construct. Therefore long term archiving, curation and dissemination of these resources needs attention in order to maximize their exploitation in R&D. On the one hand, many treebanking projects produce very useful results but their dissemination is often limited to a particular treebank or type of treebank. On

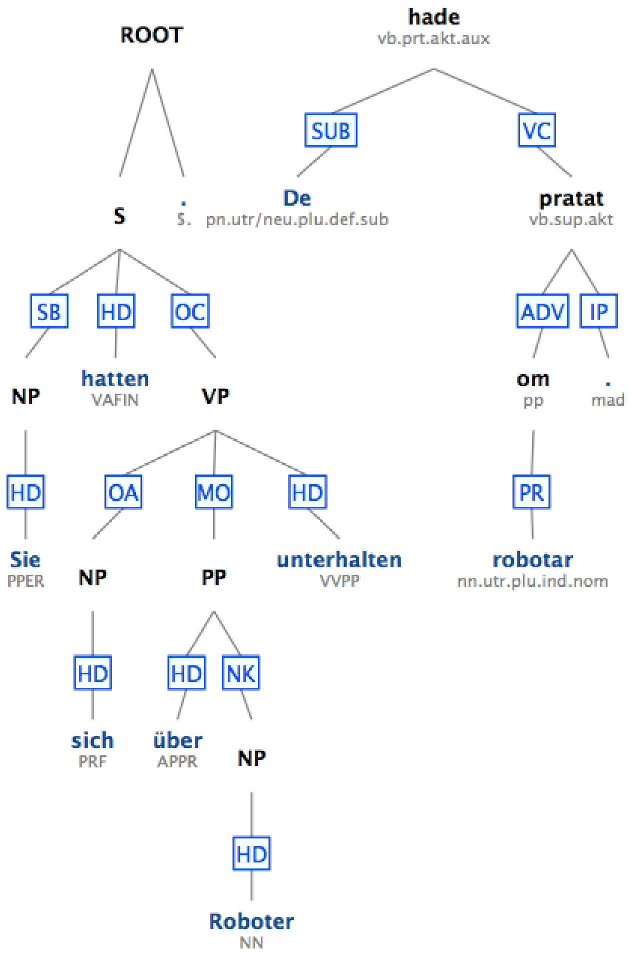


Figure 5: Parallel display of German constituency and Swedish dependency structures: *Sie hatten sich über Roboter unterhalten. / De hade pratat om robotar.*

the other hand, current open infrastructures for language resources and tools, such as META-SHARE and CLARIN, aim at building large catalogs and target large user groups, but they do not offer specific middleware or services needed for the construction and exploration of treebanks. INESS intends to fill this gap. We have in this paper presented a status report for this infrastructure.

INESS offers an expanding number of services for a steadily increasing number of treebanks, placing unprecedented emphasis on usability, on powerful search and analysis, and on advanced visualization. INESS intends to be an open infrastructure: it invites the participation of other treebanking projects and is currently negotiating with partners to set up mirrors of the infrastructure. By establishing common access, exploration and visualization of various treebank types through a uniform web-based interface, the threshold for actually using treebanks is lowered for a potentially large audience of users.

Currently, the user base of INESS is still limited. In the past year, the infrastructure has been tested mostly by internal users, and feedback has resulted in several improvements to the user interface. A more extensive, systematic user evaluation is scheduled for 2013. It is our goal that eventually, users with even a minimal linguistic background will consult treebanks in INESS almost as easily as they consult a dictionary or grammar book. We also believe that grammar teaching materials will eventually link to treebanks.

Building on our experiences so far, we envisage that INESS will soon provide web services for many treebanks, including a large treebank for Norwegian with unprecedented detail which is presently being constructed. As the infrastructure is scaling up, syntactic analysis and search must run on high-performance computing platforms in order to have an acceptable turnaround, especially when reparsing (and redisambiguating) an entire corpus with a new grammar version. Furthermore, treebanks need considerably more storage space than corpora annotated at word level only, especially when all analyses of each sentence are stored and discriminants are cached. The INESS infrastructure therefore runs on a 128-core HPC cluster using fast disk access and high-speed internal networking (cf. Figure 9). In its next phase, it will also use national eInfrastructure facilities.

Although the INESS infrastructure is fully operative, further research will allow its evolution in response to new requirements and technologies. Areas of special attention are the search and visualization middleware and the interface and user profiles. In particular, visualization of large structures is a daunting challenge. On the one hand, large screens with high resolutions are desirable physical media. On the other hand, there will be a need for continued research on innovative visualization, which may draw on experiences with visualization of large, complex structures in other fields.[9]

Access to the INESS resources and services is as yet on the basis of ad hoc usernames, while some treebanks are fully open. Current work is aimed at improved handling of licenses and metadata (in cooperation with META-NORD and CLARIN) and the integration of federated authentication and authorization, allowing users from several affiliations to log in with their local user name. Tests of federated user authentication were recently successfully concluded, using an interface to the Norwegian national FEIDE federated ID provider through a SAML 2.0 protocol (Uninett, 2010). Once a trust mechanism for authenticating users from other locations is in place, users will be able to create profiles, define preferences and store search expressions for future use. Users will also be able to upload and parse 'private' treebanks, even if sharing of treebanks is highly encouraged.

8. Acknowledgments

The research reported on in this paper has received funding from the Research Council of Norway under the National Financing Initiative for Research Infrastructure and from the ICT Policy Support Programme as part of the Competitiveness and Innovation Framework Programme, grant agreement no. 270899.

[9] E.g. Jmol for visualization of large molecules, http://www.jmol.org/.

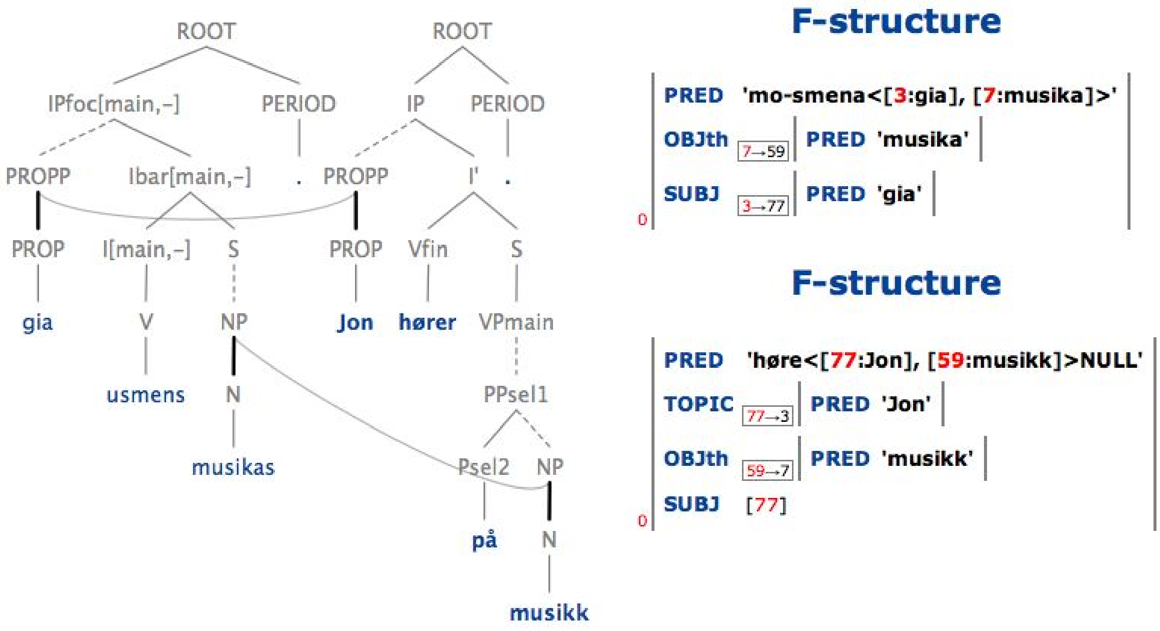


Figure 6: Parallel display of Georgian and Norwegian c-structures and f-structures

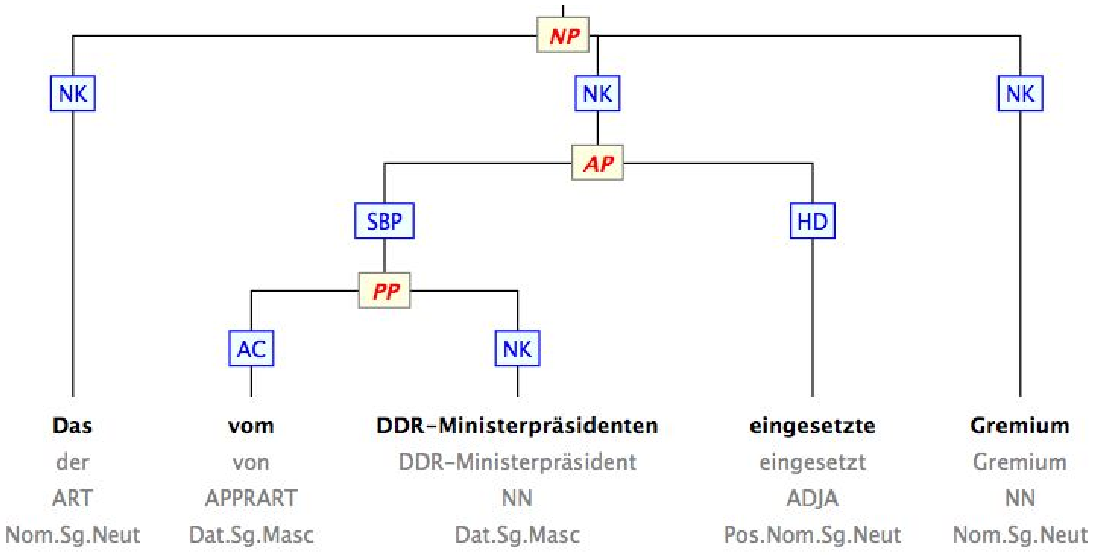
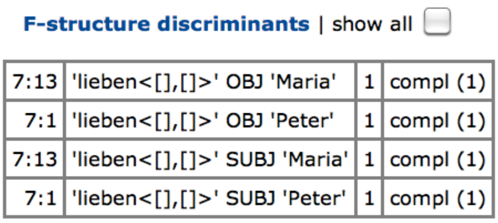


Figure 7: Search example: a solution for NP > AP > PP



7:13	'lieben<[],[]>' OBJ 'Maria'	1	compl (1)
7:1	'lieben<[],[]>' OBJ 'Peter'	1	compl (1)
7:13	'lieben<[],[]>' SUBJ 'Maria'	1	compl (1)
7:1	'lieben<[],[]>' SUBJ 'Peter'	1	compl (1)

Figure 8: Discriminants for the ambiguous German sentence *Peter liebt Maria*.

9. References

Emily M. Bender, Dan Flickinger, and Stephan Oepen. 2011. Grammar engineering and linguistic hypothesis testing: Computational support for complexity in syntactic analysis. In Emily M. Bender and Jennifer E. Arnold, editors, *Language from a Cognitive Perspective: Grammar, Usage, and Processing*, CSLI Lecture Notes 201, pages 5–29. CSLI Publications.

Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Hladká. 2003. The Prague Dependency Treebank. In Anne Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*, chapter 7, pages 103–127. Kluwer Academic Publishers.

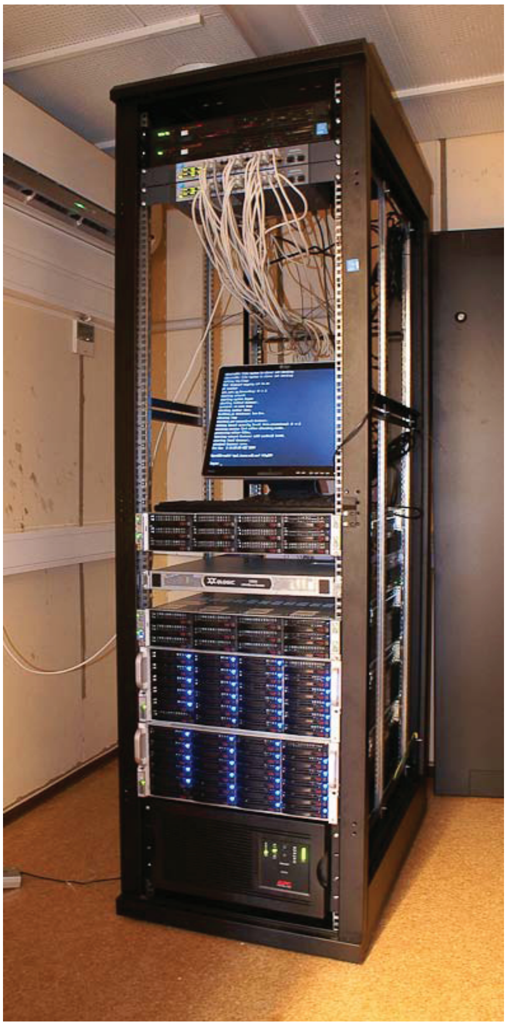


Figure 9: The INESS HPC cluster

Helge Dyvik, Paul Meurer, Victoria Rosén, and Koenraad De Smedt. 2009. Linguistically motivated parallel parsebanks. In Marco Passarotti, Adam Przepiórkowski, Sabine Raynaud, and Frank Van Eynde, editors, *Proceedings of the Eighth International Workshop on Treebanks and Linguistic Theories*, pages 71–82, Milan, Italy. EDUCatt.

Jostein Gaarder. 1991. *Sofies verden: roman om filosofiens historie*. Aschehoug, Oslo, Norway.

Jan Hajič. 1998. Building a syntactically annotated corpus: The Prague Dependency Treebank. In *Issues of Valency and Meaning*, pages 106–132. Karolinum, Praha.

Ronald M. Kaplan, Tracy Holloway King, and John T. Maxwell. 2002. Adapting existing grammars: the XLE experience. In *Proceedings of the COLING-2002 Workshop on Grammar Engineering and Evaluation, Taipei, Taiwan*.

Catherine Lai and Steven Bird. 2004. Querying and updating treebanks: A critical survey and requirements analysis. In *Proceedings of the Australasian Language Technology Workshop*, pages 139–146.

Wolfgang Lezius. 2002. TIGERSearch – Ein Suchwerkzeug für Baumbanken. In Stephan Busemann, editor, *Proceedings der 6. Konferenz zur Verarbeitung natürlicher Sprache (KONVENS 2002), Saarbrücken*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

John Maxwell and Ronald M. Kaplan. 1993. The interface between phrasal and functional constraints. *Computational Linguistics*, 19(4):571–589.

Paul Meurer. 2012 (forthcoming). INESS-Search: A search system for LFG (and other) treebanks. In *Proceedings of the LFG '12 Conference*.

Joakim Nivre, Koenraad De Smedt, and Martin Volk. 2005. Treebanking in Northern Europe: A white paper. In Henrik Holmboe, editor, *Nordisk Sprogteknologi 2004. Årbog for Nordisk Sprogteknologisk Forskningsprogram 2000-2004*, pages 97–112. Museum Tusculanums Forlag, Copenhagen.

Stephan Oepen and Daniel P. Flickinger. 1998. Towards systematic grammar profiling. Test suite technology ten years after. *Journal of Computer Speech and Language*, 12(4):411–436.

Victoria Rosén, Paul Meurer, and Koenraad De Smedt. 2005. Constructing a parsed corpus with a large LFG grammar. In *Proceedings of LFG'05*, pages 371–387. CSLI Publications.

Victoria Rosén, Paul Meurer, and Koenraad De Smedt. 2007. Designing and implementing discriminants for LFG grammars. In Tracy Holloway King and Miriam Butt, editors, *The Proceedings of the LFG '07 Conference*, pages 397–417. CSLI Publications, Stanford.

Victoria Rosén, Paul Meurer, and Koenraad De Smedt. 2009. LFG Parsebanker: A toolkit for building and searching a treebank as a parsed corpus. In Frank Van Eynde, Anette Frank, Gertjan van Noord, and Koenraad De Smedt, editors, *Proceedings of the Seventh International Workshop on Treebanks and Linguistic Theories (TLT7)*, pages 127–133, Utrecht. LOT.

Uninett. 2010. Feide integration guide: Integrating a service provider with Feide. Technical Report version 1.3, Uninett, Trondheim, August.

Gertjan van Noord. 2009. Huge parsed corpora in LASSY. In Frank Van Eynde, Anette Frank, Koenraad De Smedt, and Gertjan van Noord, editors, *Proceedings of the Seventh International Workshop on Treebanks and Linguistic Theories (TLT7)*, pages 115–126. LOT.

Joel Wallenberg, Anton Karl Ingason, Einar Freyr Sigurðsson, and Eiríkur Rögnvaldsson. 2011. Icelandic Parsed Historical Corpus (IcePaHC) version 0.9.

A German-Georgian Parallel Treebank Project

Oleg Kapanadze

OK’OMPLEX – Innovative Information and Language Technologies
V. Beridze, 1
0118 Tbilisi, Georgia
E-mail: ok@caucasus.net

Abstract

This paper reports about efforts on building a parallel treebank for a typologically dissimilar language pair, namely German and Georgian. The project aims at supporting interdisciplinary collaboration in the field of jurisprudence adding a Natural Language Technology (NLT) angle to the human translation issue. The objective of the project is development of a bilingual Treebank which will be based on a linguistically annotated and syntactically parsed German-Georgian parallel legislative text corpus.

1. Introduction

Parallel corpora are language resources that contain texts and their translations, where the texts, paragraphs, sentences, and words are linked to each other. In the past decades they became useful not only for NLP applications, such as machine translation and multi-lingual lexicography, but are considered also very useful for empirical language research in contrastive and translation studies.

Naturally-occurring text in many languages are annotated for linguistic structure. A **Treebank** is a text corpus in which each sentence has been annotated with syntactic structure. *Treebanks* are often created on top of a corpus that has already been annotated with part-of-speech tags. The annotation can vary from constituent to dependency or tecto-grammatical structures. In turn, *Treebanks* are sometimes enhanced with semantic or other linguistic information and are skeletal parses of sentences showing rough syntactic and semantic information.

Treebanks have become valuable resources as repositories for linguistic research. They can be used in *translation studies*, in corpus linguistics for studying syntactic phenomena, in computational linguistics as evaluation corpora for different NLT systems or for training and testing parsers.

In this paper, we describe our work on building a parallel Treebank for a typologically dissimilar language pair, namely German and Georgian, where the sentences in each language involved are syntactically analyzed, and sentences and words are aligned.

2. Data for experiment

For the low-density languages, including Georgian, parallel texts are very rare, if at all existent. The parallel corpus used for this study comprises German texts and their translations into Georgian language compiled for the GREG project (Kapanadze et al., 2002, Kapanadze, 2010). The GREG lexicon itself contains valency data with the manually aligned Georgian, Russian, English and German verbs (ca. 1250) that are augmented with examples of sentences considered as translation equivalents. Each subcorpus used for the study has a size of roughly 2623 sentence pairs that correspond to different syntactic subcategorization frames considered as German-Georgian translation equivalents.

3. Building Monolingual Treebanks

3.1. Morphological analysis

Georgian is an agglutinative language using both suffixing and prefixing. For the Georgian text analyses has been applied a finite-state morphological transducer using the XEROX FST tools (Kapanadze 2010a,b), (Kapanadze 2009). The Georgian FST transducer utilizes a number of the formalisms supported by the XEROX toolkit (Beesley and Karttunen, 2003). The lexicon specification language *lexc* was used for modeling the lexicon and for constraining the morphotactics. It consists of 7 modules for noun, adjective, pronoun, numeral, adverb, verb and the minor categories analysis. Currently there are two versions of the Georgian FST transducer available in the MS Windows platform and in the LINUX UBUNTU version.

3.2 Syntactic parsing

The syntactic annotation employs parts-of-speech, morphological properties, and dependency functions. Every sentence is assumed to have a unique head and all other tokens, except punctuation marks, are direct or indirect dependents of the head. Monolingual files are XML-formatted.

Using the morphologically annotated bilingual corpus and the GREG lexicon data, both the German and the Georgian texts were syntactically annotated manually. For this purpose is used *the Synpathy,* a tool for syntactical annotation developed at Max Plank Institute for Psycholinguistics, Nijmegen, the Netherlands (www.mpi.nl/corpus/manuals/manual-synpathy.pdf).

The German treebank annotation follows the TIGER annotation scheme (Skut et al., 1997, Brants et al., 2002). The Georgian treebank was annotated according an adapted version of the German TIGER guidelines with the necessary changes relevant to the Georgian grammar formal description. The output of the syntactic annotation is in the TIGER-XML format. From the TIGER-XML format, the syntactic annotation may be visualized with tools like TIGER Search, representing dependency graphs for sentences in German and in Georgian as shown in Figure 1 and Figure 3.

The monolingual treebanks converted into TIGER-XML, are a powerful database-oriented representation for graph structures. In a TIGER-XML graph each leaf (= token) and each node (= linguistic constituent) has a unique identifier (Samuelsson and Volk, 2007). We use these unique identifiers for the phrase and word alignment across trees in corresponding translation units.

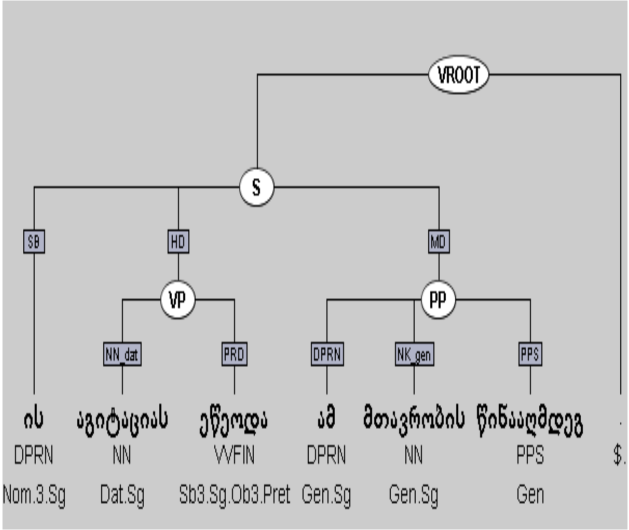


Figure1: A screenshot of an annotated Georgian sentence.

An XML representation is also used for storing this alignment. In the Figure 2 and Figure 4 there are representations of the sentences from the Figure 1 and the Figure 3 in the TIGER-XML format.

```
<body>
<s id="s12">
  <graph root="s12_502" discontinuous="true">

  <terminals>
      <t id="s12_1" word="ის" pos="DPRN"
        morph="Nom.3.Sg" />
      <t id="s12_2" word="აგიტაციას" pos="NN"
        morph="Dat.Sg" />
      <t id="s12_3" word="ეწეოდა" pos="VVFIN"
        morph="Sb3.Sg.Ob3.Pret" />
      <t id="s12_4" word="ამ" pos="DPRN"
       morph="Gen.Sg" />
      <t id="s12_5" word="მთავრობის" pos="NN"
        morph="Gen.Sg" />
      <t id="s12_6" word="წინააღმდეგ" pos="PPS"
       morph="Gen" />
       <t id="s12_7" word="." pos="$." morph="--" />
 </terminals>

 <nonterminals>
   <nt id="s12_502" cat="S">
        <edge label="SB" idref="s12_1" />
        <edge label="HD" idref="s12_503" />
        <edge label="MD" idref="s12_501" />
   </nt>
   <nt id="s12_503" cat="VP">
        <edge label="NN_dat" idref="s12_2" />
        <edge label="PRD" idref="s12_3" />
   </nt>
   <nt id="s12_501" cat="PP">
        <edge label="DPRN" idref="s12_4" />
        <edge label="NK_gen" idref="s12_5" />
        <edge label="PPS" idref="s12_6" />
   </nt>
```

```
<nt id="s12_VROOT" cat="VROOT">
      <edge label="--" idref="s12_502" />
      <edge label="--" idref="s12_6" />
  </nt>
</nonterminals>
</graph>
</s>
```

Figure 2: A TIGER-XML format representation of a Georgian sentence from the Figure 1.

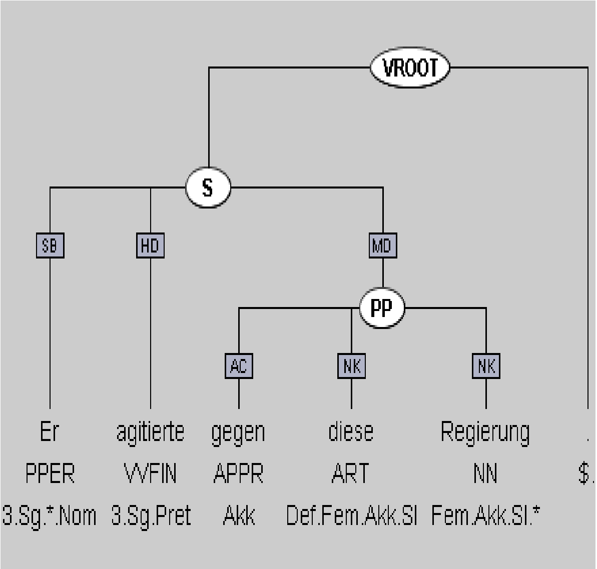


Figure3: A screenshot of the corresponding annotated sentence in German language.

```
<body>
<s id="s12">
  <graph root="s12_13" discontinuous="true">

<terminals>
     <t id="s12_1" word="Er" pos="PPER"
      morph="3.Sg.*.Nom" />
     <t id="s12_2" word="agitierte" pos="VVFIN"
      morph="3.Sg.Pret" />
     <t id="s12_3" word="gegen" pos="APPR"
     morph="Akk" />
     <t id="s12_4" word="diese" pos="ART"
     morph="Def.Fem.Akk.Sl" />
     <t id="s12_5" word="Regierung" pos="NN"
     morph="Fem.Akk.Sl.*" />
     <t id="s12_6" word="." pos="$." morph="--" />
</terminals>
<nonterminals>
    <nt id="s12_500" cat="CS">
    </nt>
     <nt id="s12_502" cat="S">
         <edge label="SB" idref="s12_1" />
          <edge label="HD" idref="s12_2" />
          <edge label="MD" idref="s12_501" />
     </nt>

   <nt id="s12_501" cat="PP">
          <edge label="AC" idref="s12_3" />
          <edge label="NK" idref="s12_4" />
          <edge label="NK" idref="s12_5" />
          <edge label="NK" idref="s12_6" />
        </nt>

  <nt id="s12_VROOT" cat="VROOT">
        <edge label="--" idref="s12_502" />
        <edge label="--" idref="s12_6" />
   </nt>
  </nonterminals>
  </graph>
 </s>
```

Figure 4: A TIGER-XML format representation of a German sentence from the Figure 3.

4. Building a Parallel Treebank

Alignment of a Monolingual German and a Monolingual Georgian Treebanks into a Parallel Treebank

This procedure is done with help of *the Stockholm TreeAligner*, a tool for work with parallel treebanks which inserts alignments between pairs of syntax trees (Samuelsson and Volk, 2005, Samuelsson and Volk, 2006). The Stockholm TreeAligner handles alignment of tree structures, in addition to word alignment, which – according to its developers - is unique (Samuelsson and Volk, 2006).

Phrase alignment can be regarded as an additional layer of information on top of the syntax structure. It shows which part of a sentence in the German language is equivalent to which part of a corresponding sentence in the Georgian language. This is done with the help of a graphical user interface of the Stockholm TreeAligner. We drew alignment lines manually between pairs of sentences, phrases and words over parallel syntax trees. Figure 5 shows a screenshot with two aligned trees from Figure 1 and Figure 3. We intended to align as many phrases as possible. The goal is to show translation equivalence. Phrases shall be aligned only if the tokens, that they span, represent the same meaning and if they could serve as translation units outside the current sentence context. The grammatical forms of the phrases need not fit in other contexts, but the meaning has to fit.

The Stockholm TreeAligner guidelines allow phrase alignments within m : n sentence alignments and 1 : n phrase alignments. Even though m : n phrase alignments are technically possible, we have only used 1 : n phrase alignments, for simplicity and clarity reasons. One example of 1: n alignment on the word level is the Georgian multi-word expression for “აგიტაციის გაწევა” represented under a VP node in the Figure 1, which is one word (“agitierte”) in the corresponding German sentence from the Figure 3. The 1 : n alignment option is not used if a node from one tree is realized twice in the corresponding tree, e.g. a repeated subject in coordinated sentences.

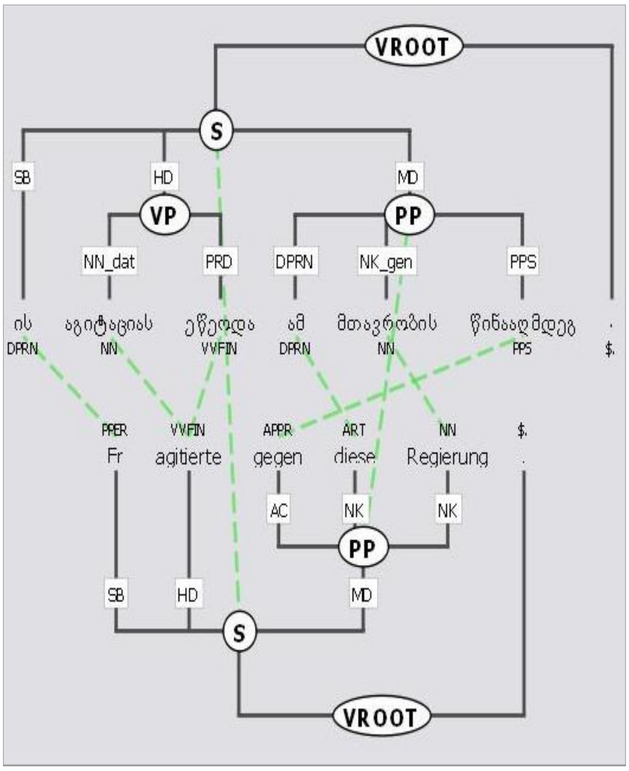


Figure 5: A screenshot with aligned trees from Figure 1 and Figure 3.

The designers of the Stockholm TreeAligner differentiate between two types of alignment, displayed by different colours. Nodes and words representing exactly the same meaning are aligned as exact translation correspondences using the green colour for lines as it is shown on the Figure 6. In this regard a German word (“agitierte”) alignment to the Georgian Verb Phrase “აგიტაციის გაწევა” as an exact one, might be considered problematic. Nevertheless, in such a case a prerequisite for this solution is that they could serve as translation units outside the current sentence context.

If nodes and words represent just approximately the same meaning, they are aligned as fuzzy translation correspondences by means of lines in the red colour as it is shown in the Figure 7 bellow.

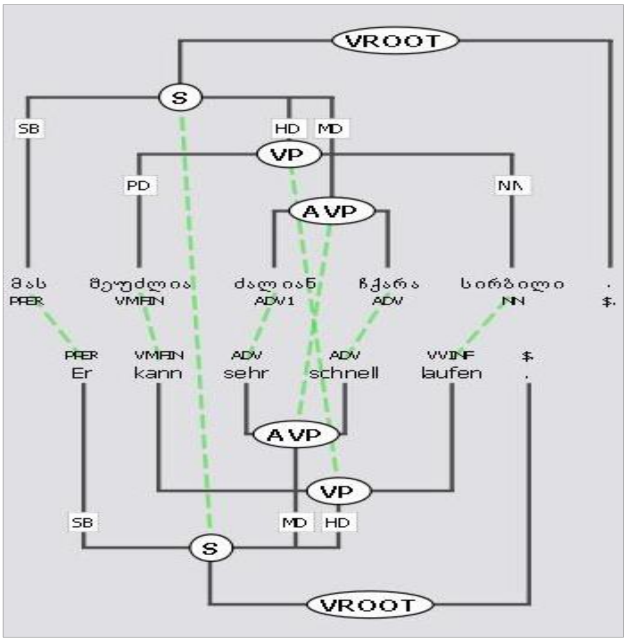


Figure 6: A screenshot of the TreeAligner of the Georgian and German sentences with 1:1 aligned words and phrases.

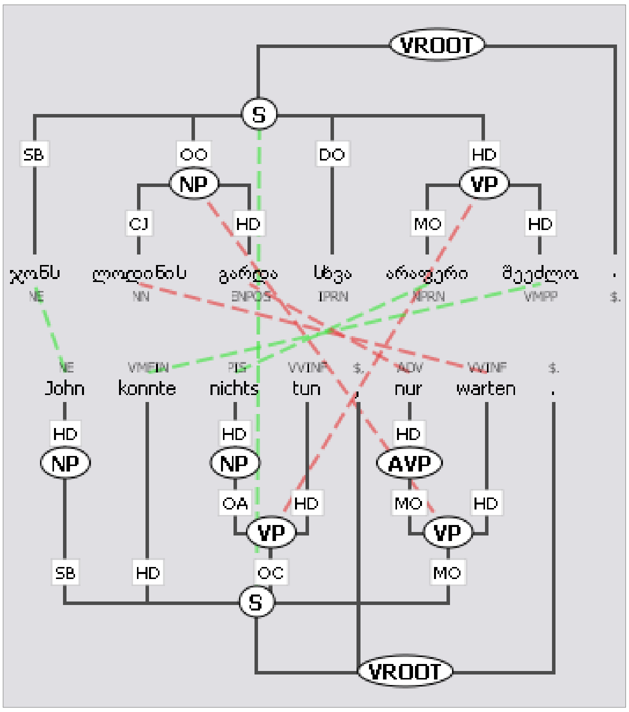


Figure 7: A screenshot of the TreeAligner of the Georgian and German sentences with exact and fuzzy aligned words and phrases.

In an appendix an example of an annotated compound Georgian and German sentences with exact and fuzzy alignment on simple clause and phrase level could be viewed.

5. Conclusions and Future Research

At the initial phase of presented experiment we made an overview of experience in building parallel threebanks for languages with different structures (Megyesi and Dahlqvist, 2007), (Megyesi et al., 2006), (Grimes et al., 2011), (Rios et al., 2009).

As it is reported in a Quechua-Spanish parallel treebank project, due to strong agglutinative structure of Quechua language, it was decided to annotate the Quechua treebank on morphemes rather than words. This allowed the authors to link morpho-syntactic information precisely to its source. Besides, building phrase structure trees over Quechua sentences does not capture the characteristics of the language. Therefore, they have chosen Role and Reference Grammar. By using nodes, edges and secondary edges in the Stockholm annotation tool they were able to represent the most important aspects of Role and Reference syntax for Quechua sentences (Rios et al. 2009).

Although the Georgian language is also an agglutinative language with suffixing and prefixing, there is no need to annotate the Georgian Treebank on morphemes. However, for syntactic annotation in the Georgian language a precise description of a specific structure/mechanism of its clause is necessary. "The Georgian clause is a word collocation which draws on coordination and government of the linked verb and noun sequence" [Chikobava, 1928]. The types of syntactic relations in the Georgian clause differ significantly from that observed in the Indo-European or in other languages. In the English Language there are just a small number of verbs that govern the nouns linked to them as indirect actants and demand those nouns to stand in an indirect case form (e.g. *John believes* ***him*** *to be innocent*). Besides, the actants involved do not induce changes in the verb form. In contrary, in the polyvalent Georgian verb the actants are marked with specific affixes in a verb. The most significant difference from the structure of the Indo-European syntactic relations model is that in the Georgian clause we have a mutual government and agreement relations or a bilateral coordination phenomenon between verb-predicate and noun-actants which number may reach up to three in a single clause. It anticipates control of the noun case forms by verbs, whereas the verbs, in their turn, are governed by nouns with respect to a grammatical person. Therefore, according to [Chikobava, 1928] in a syntactic description of Georgian the concepts of a *Major* and a *Minor Coordinate*, instead of *Subject* and *Object*, are preferable. Moreover, in the verb forms of a certain semantic type an indirect object has preference as a *Major Coordinate* over a *Subject* (a *Minor coordinate*) in the respect of its marking in a verb form. Nevertheless, unlike the Quechua language, Georgian syntax can be sufficiently well represent by means of dependency relations and there is no need to utilize a different approach to capture the Georgian language structural peculiarities.

As it has been already mentioned, parallel texts for the Georgian language are very rare. Nevertheless, in the process of experimental undertakings for building a German-Georgian Parallel Treebank, we have discovered a text repository comprising the German-Georgian parallel texts in jurisprudence. This bilingual text corpus is a collection of the Georgian laws translated into the German language. The corpus is created by human translators thanks to the GTZ (*Gesellschaft für Technische Zusammenarbeit*) and German jurists that supported development of a modern Georgian legislation during almost two decades.

In the presented experiment, except morphological analysis for the Georgian text, syntax structures for both languages we have created completely manually. For further expending the parallel German-Georgian Treebank for legislative texts we intend to experiment with the Tree-to-Tree (t2t) Alignment Pipe (Killer, Sennrich and Volk, 2011a, b). This is a collection of python scripts, generating automatically aligned parallel treebanks from multilingual web resources or existing parallel corpora. The pipe contains wrappers for a number of freely available NLP software programs used for tokenization (NLTK Treebank), sentence alignment (Hunalign, Vanila Aligner, Microsoft BSA), word alignment (Giza++), syntactic parsing (Stanford Parser, Berkley Parser) and Tree to Tree Aligner (Zhechev 2009).

On the final stage the Tree-to-Tree alignment pipe prepares an input for its further processing in the Graphical User Interface of the Stockholm TreeAlligner which will be used for the aligned German-Georgian Treebank visualization and correction procedures.

References

Beesley K. R. and L. Karttunen. (2003). Finite State Morphology. CSLI Publications.

ჩიქობავა, ა. (1928). მარტივი წინადადების პრობლემა ქართულში, თბილისი. [Chikobava A. (1928). The Problem of the Simple Sentence in Georgian. Tbilisi].

Grimes S., Li, X., Bies A., Kulick S. Ma, X. and S. Strassel. (2011). Creating Arabic-English Parallel Word-Aligned Treebank Corpora at LDC. Proceedings of the Second Workshop on Annotation and Exploitation of Parallel Corpora. The 8th International Conference on Recent Advances in Natural Language Processing (RANLP 2011). Hissar, Bulgaria. 2011.

Kapanadze O., Kapanadze N., Wanner L., and St. Klatt. (2002). Towards A Semantically Motivated Organization of A Valency Lexicon for Natural Language Processing: A GREG Proposal. Proceedings of the EURALEX conference, Copenhagen.

Kapanadze O. (2010a). Verbal Valency in Multilingual Lexica. In: Workshop Abstracts of the 7th Language Resources and Evaluation Conference-LREC2010. Valletta, Malta.

Kapanadze O. (2010b). Describing Georgian Morphology with a Finite-State System. In A. Yli-Jura et al. (Eds.): Finite-State Methods and Natural Language Processing 2009, Lecture Notes in Artificial Intelligence, Volume 6062, pp.114-122, Springer-Verlag, Berlin Heidelberg .

Kapanadze O. (2009). Finite State Morphology for the Low-Density Georgian Language. In: FSMNLP 2009 Pre-proceedings of the Eighth International Workshop on Finite-State Methods and Natural Language Processing. Pretoria, South Africa

Killer M., Sennrich R. and M. Volk (2011). From Multilingual Web-Archives to Parallel Treebanks in Five Minutes. In Multilingual Resources and Multilingual Applications. Proceedings of the Conference of the German Society for Computational Linguistics and Language Technology (GSCL 2011).

Megyesi B. and B. Dahlqvist. (2007). A Turkish-Swedish Parallel Corpus and Tools for its Creation. In Proceedings of Nordiska Datalingvistdagarna (NoDaL-iDa 2007).

Megyesi B., Hein A.S. and E. C. Johanson. (2006). Building a Swedish-Turkish Parallel Corpus. In Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006).

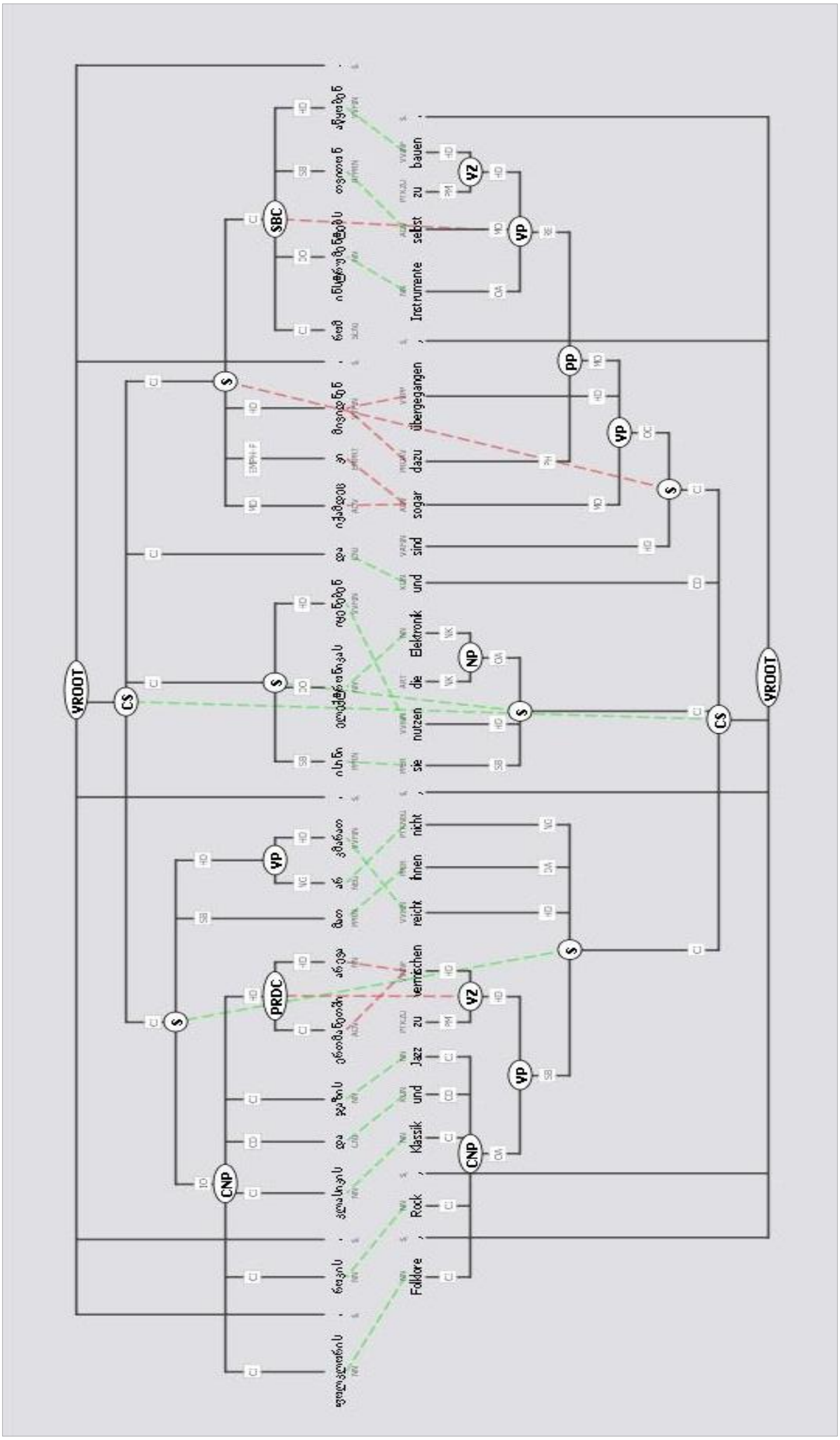
Rios A., Göhring A. and M. Volk. (2009). Quechua-Spanish Parallel Treebank. In: 7th Conference on Treebanks and Linguistic Theories, Groningen, 2009.

Samuelsson Y. and M. Volk. (2005). Presentation and Representation of Parallel Treebanks. In Proceedings of the Treebank-Workshop at Nodalida, Joensuu, Finland.

Samuelsson Y. and M. Volk. (2006). Phrase Alignment in Parallel Treebanks. In Proceedings of 5th Workshop on Treebanks and Linguistic Theories, Prague, Czech Republic.

Samuelsson Y. and M. Volk. (2007). Alignment Tools for Parallel Treebanks. In GLDV Frühjahrstagung, Tübingen, Germany, 2007.

Zhechev V. (2009). Automatic Generation of Parallel Treebanks. An Efficient Unsupervised System. Dissertation.
http://www.ventsislavzhechev.eu/Home/Publications-files/PhD%20Thesis%20Zhechev20%E2%80%94%C2%A0Final.pdf.


VROOT
CS
S
CNP
PRDC
VP
VZ
SBC
NP
PP
Folklore
,
Rock
,
Klassik
und
Jazz
zu
vermischen
reicht
ihnen
nicht
,
sie
nutzen
die
Elektronik
und
sind
sogar
dazu
übergegangen
,
Instrumente
selbst
zu
bauen
.

Czech treebanking unlimited

Tomáš Jelínek, Vladimír Petkevič, Alexandr Rosen and Hana Skoumalová

Charles University, Faculty of Arts, Institute of Theoretical and Computational Linguistics
Celetná 13, 110 00 Praha 1, Czech Republic
{tomas.jelinek|vladimir petkevic|alexandr.rosen|hana.skoumalova}@ff.cuni.cz

Abstract

We build a large treebank of Czech, avoiding manual effort by using a parser, supplemented by a rule-based correction tool. A potentially underspecified morphological and syntactic annotation scheme offers multiple visualisation and export options, customizable in shape and detail according to the preferences of humans or computer applications. The annotation scheme consists of three layers: graphemics, morphology and constituency-based syntax, and is supported by a lexicon (with a morphological, multi-word and syntactic part) and a grammar. Annotation on any of the interlinked layers can be missing; ambiguous or undecidable phenomena are represented by underspecification and distributive disjunction.

1. Introduction

Treebanks are often built with considerable manual effort and cannot match the size of other text corpora. Yet whenever a language is lucky to have one, human users and applications alike benefit from its existence.[1] At first, the size of the corpus, the style and theoretical background of the annotation, or even its detail are not an issue. But sooner or later the users realize that at least for some tasks a treebank should reach higher volumes.[2] Additionally, various wishes concerning a "proper" annotation scheme are voiced, often reflecting the division of syntactic theory into a number of camps. Finally, some data are difficult to annotate unambiguously and call for underspecified description (Oliva, 2001). We wish to propose answers to these issues.

The bigger the better, but not at an unbearable decrease in reliability. The largest treebanks reach the modest sizes of several million words. To match the size of a balanced POS-tagged corpus, the use of parsing tools is inevitable. But parsers still perform less reliably than POS taggers and the cost of manual checking is prohibitive.

Building on previous efforts in treebank annotation of Czech, especially the Prague Dependency Treebank – PDT (Hajič, 2006, i.a.), we combine a stochastic parser (Holan and Žabokrtský, 2006) with a rule-based correction module (Jelínek, 2012), diminishing the parser's error rate. The goal is to provide syntactic annotation for all contemporary written texts in the Czech National Corpus.[3] The texts represent a balanced mix of genres, with the total of 1.3 billion tokens at the moment.

We are not alone in realizing the usefulness of "parsebanks". In addition to those serving as a testbed for a specific rule-based grammar and linguistic theory, such as LinGO Redwoods,[4] there are other large automatically parsed corpora available, such as LASSY,[5] with Dutch texts numbering 1.5 billion tokens. The higher error rate does not seem to be a serious obstacle for tasks such as automatic valency acquisition.

An additional way to make the automatically parsed corpus useful is to offer fully customizable visualisation and export, including various folding and filtering options. Then the results are potentially less sensitive to errors in details or in more embedded structures. The scheme can also represent inherent ambiguities (in principle impossible to resolve even in a wider context), and allow for uncertainty or underspecification, potentially remediable later.

Even more importantly, there are various modes of displaying syntactic structure, according to the preference of the user, who may subscribe to a specific syntactic theory and may be put off by theoretical bias of a treebank. Yet despite differences in appearance and focus, all linguistic theories strive to describe the same object – a natural language. There is a large pool of implicit wisdom shared by all syntactic theories and a significant overlap of knowledge can be extracted from all theory-specific formats.

This multitude of syntactic paradigms can be approached by introducing parallel annotation layers (Hautli et al., 2012). However, we opt for a more flexible scheme, which is feasible precisely because all theories have to cope with the same issues. We propose a treebank offering different views of syntactic annotation while based on a single core pattern. In addition to constituency and dependency trees of various shapes, suited to the taste of linguistic experts, some views may be appealing to a wider audience of students and professionals dealing with language.

With these aims in mind, our explicitly defined annotation scheme consists of a potentially underspecified morphological and syntactic core, complemented by multiple interaction shells, customizable in shape and detail according to the preferences of humans or computer applications.

In Section 2. we explore the annotation scheme in more de-

[1] There is ample evidence of the usefulness of treebanks, even of those concerning less widely spoken languages, such as Czech, cf. `http://ufal.mff.cuni.cz/pdt2.0/`.

[2] This may be a concern for experts in machine learning and linguistic theoreticians alike. The latter are sometimes interested in very specific and rare cases to support a theoretical claim. Due to the lack of such evidence in treebanks of a smaller size, they resort to elicitation and/or use a larger corpus without syntactic annotation (Przepiórkowski and Rosen, 2005).

[3] The "SYN" family of subcorpora, see `http://www.korpus.cz/english/struktura.php`.

[4] `http://lingo.stanford.edu/redwoods/`

[5] `http://www.let.rug.nl/~vannoord/Lassy/`

tail, before discussing the conversion from the dependency trees produced by the parser to our format in Section 3. and the rule-based correction module, applied to the parser's output, in Section 4.

2. The core structure

The texts are annotated by a single core scheme, interpretable in different ways. The syntactic structure part is based on constituency, which supports structural ambiguity and underspecification more easily and can serve as a source from which multiple representations can be derived (see Section 2.3.). The annotation is licensed by a formal description, a de facto grammar, which additionally facilitates format conversions (see Section 2.2.).

2.1. Separation of graphemics, morphology and syntax

Word order and syntactic structure are represented in the core structures as formally distinct dimensions to support the choice of similarly separate or integral visualisation and comparison. In fact, each sentence is represented at three inter-linked levels: graphemics (orthographic words), morphology (syntactic words), and syntax (trees). The level of graphemics allows for handling contractions and similar purely orthographical phenomena. Reflexives subject to haplology are restored 2.1., and contractions such as *ses*, represented as a single graphemic unit, are analysed as two morphological forms: here as a reflexive pronoun/particle and the 2nd person auxiliary. More mismatches in the number of tokens occur between the levels of morphemics and syntax, where punctuation is omitted.

(1) Rozhodl se umýt.
decided$_{masc,sg}$ REFL wash$_{inf}$
'He has decided to wash himself.'

The haplologized item *se* is both a reflexive particle, a part of an inherent reflexive *rozhodl se*, and a reflexive pronoun as the object of the transitive verb *umýt se*. As such, it is represented as two tokens on the level of morphemics:

GRAPHEMICS	rozhodl	se		umýt
MORPHEMICS	*lpple,masc,sg*	*pcle*	*prnrefl,acc*	*inf*

The two interpretations of *se* appear as two nodes in the syntactic structure below. The boxed constituent stands for the inherently reflexive verb as a multiword.

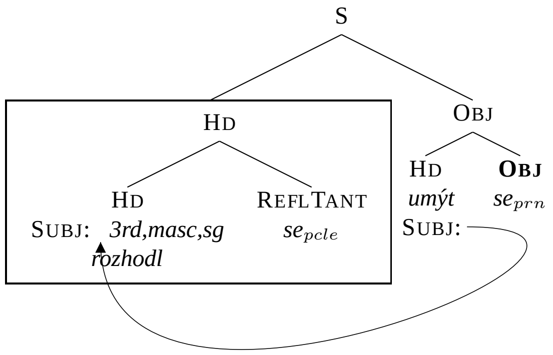


Figure 1: Haplology in a tree

Mismatches in the number of nodes at the individual levels (as in the case of *se* above) are kept at a minimum, elided items of all sorts are not restored as separate nodes but recorded in the node-internal structure of their heads or referring expressions as arguments, adjuncts or antecedents. E.g., in Fig. 1, PRO is represented equivalently as a link between the subject of the infinitive and the subject of the finite verb. All such phenomena are represented by linking the infinitive, predicative complement, base coordinated verb etc. across the structure with its argument. The link is labeled by the relevant syntactic function.

Links of another type make sure that agreeing categories in subject-predicate or adjective-noun agreement structures share identical values and the agreeing forms are identified. In the linear display (2), agreeing forms are underlined. Such links are predictable from syntactic structure and functional labels, and are inferred using the formal description of the annotation (grammar and lexicon).

Depending on the choice of the user, discontinuous (non-projective) structures can be represented as such, with crossing branches of the syntax tree, or made continuous (projective) on the syntactic level, with the order of the terminal nodes different from the lower levels. The parser is able to identify non-projectivity in the assumed dependency structures, and its results are subject to checks and modifications by correction rules.

2.2. Formal description of the annotation

To enforce consistency in the annotated data and to support interaction with the annotation, all syntactic structures in the corpus have to be licensed by a formal grammar. This includes a requirement that words and constituents have their appropriate (potentially underspecified) sets of features. A lexicon, compiled from existing resources and the corpus, and coupled with the grammar, is used to index word tokens using lemmas with appropriate categories, as well as compound forms and multi-word lexical units.

The grammar, based on the formalism of HPSG (Pollard and Sag, 1994), consists of (i) a definition of formal objects used to represent words and constituents, together with their properties, and (ii) constraints on the setup of larger objects, such as phrases, consisting of other objects. The formalism of HPSG allows for an arbitrary level of specificity of the description, and for including information of all kinds, which is useful for two reasons: a) The description should cover all structures and phenomena in the treebank, which is a hard task for a standard rule-based grammar. We need to relax some constraints or leave them unspecified, allowing for a prospect of a long-term grammar development. b) The core structures should be interpretable in various ways and according to various linguistic theories, relying on information in the structures.

Although the data in the corpus have a three-layered structure technically, they are compatible with the HPSG data format, where a single object provides all information about any expression – word or phrase. The information from the three levels, from the grammar and from the lexicon is merged and presented in a single virtual object.

In addition to checking the treebank's consistency, the grammar with the lexicon is useful (i) for inferring agreement relations, which do not have to be present in the input or supplied manually, (ii) as a base for describing format conversions to customized visualisation and export, and (iii) to supply additional annotation, such as parallel interpretations in prototypical cases of syntactic ambiguity.

The lexicon consists of three parts: morphological (specifying inflection paradigms of individual lemmas), syntactic (specifying their syntactic properties) and a lexicon of multi-word units, identifying analytical verb forms, potentially discontinuous collocations and phrasemes.

The setup of lexical resources and their relation to the three layers of annotation is shown in Fig. 2. Details about the tricky case of haplology are illustrated in Fig. 3, where tokens on the layer of graphemics are linked with tags on the morphological layer (such as `TT`... for reflexive particle, or `P7-X4`... for reflexive pronoun), which, in turn, are linked to lemmas in the morphological lexicon, and to nodes on the syntactic layer. The nodes point to the syntactic lexicon, including, e.g., info on valency, and – where appropriate – also to the multi-word lexicon.

2.3. Multiple options to display syntactic structure

The syntactic annotation scheme is designed to offer different views of its content, including constituency or dependency trees with a customizable level of abstraction (concerning, i.a., deep or surface dependencies, interpretation of function words, and identification of complex verb forms including inherent reflexives), and visualized in a horizontal or vertical mode with an arbitrary amount of detail, not necessarily by tree graphs. A linear display identifying the major (possibly discontinuous) constituents of a clause could be the option of choice for many users, see (2).

(2) A linear display of elementary syntactic structure:[6]
Ty by *se*s byl ušpinil.

Displayed information need not be explicitly encoded in the core structure, e.g., standard syntactic categories such as S, NP or PP are derived from the POS of the word or of the constituent's head and its valency.

An important side effect of less detailed visualisation is that some annotation errors can remain hidden. The parser produces about twice as many errors in syntactic structure than in syntactic functions, while the distributions of errors is uniform across all branches within a tree. By displaying only partial syntactic analysis, e.g., by using a combination of filters on main clause constituents, constituent labels, syntactic functions or syntactic structure, the number of errors visible by the user may be significantly reduced.

2.4. Surface and deep structure

Every constituent has a type – headed or unheaded – and a syntactic function. The list of types and functions is presented in Tables 1 and 2 below.

As Table 2 shows, a head can be distinguished as surface or deep; a function word such as preposition or verbal auxiliary is labeled as surface head while its sister is the deep head.[7] This allows for extracting both surface and deep dependencies from a single structure, see Fig. 4. Coordination and similar constructions are treated as headless (they are of the type UnHeaded).

Label	Description
Headed	Headed type
UnHeaded	Unheaded type with three subtypes:
– Coord	coordination structure
– Adord	adordination structure
– UnSpec	unspecified: other type of structure

Table 1: Types of constituents

Label	Description
SHd	Surface head
DHd	Deep head
Hd	Head (simultaneously surface and deep)
Subj	Subject
ObjAdvb	Object or Adverbial with two subtypes:
– Obj	Object
– Advb	Adverbial
Attr	Attribute
VbAttr	Verbal complement
ReflTant	Reflexive particle
DeAgent	Reflexive particle with deagentive meaning
Apos	Apposition
Indep	Independent constituent (parenthesis, noun in the vocative, etc.)
Memb	Member of one of the unheaded subtypes

Table 2: Syntactic functions

(3)

Ty	by	ses	byl	ušpinil.
you	would	REFL+AUX$_{2nd,sg}$	be$_{pple}$	get dirty$_{pple}$

'You would have got dirty.'

The three structures in Fig. 4 are all possible renderings of a single analysis of (3). The constituent structure has function labels for subject, object, head, surface head and deep head, and it is followed by the derived surface and deep dependency structures.[8] Complex verb forms are highlighted by boldface, contractions by a box.

It is relatively straightforward to distinguish the three types of head, and thus the shape of the surface and deep dependency structure. Lexemes identifiable in a proper syntactic context as surface heads are labeled with specific syntactic functions by the parser and form a closed class. This distinction, implying the assignment of functional labels to other nodes in the vicinity, is performed by rules operating during the conversion of the parser output.

2.5. Ambiguity and partial information

Most corpora are annotated in an unambiguous way. Yet ambiguity is sometimes inevitable for fundamental reasons,

[6] The intended meaning of the text attributes is as follows: **subject**, predicate, *object*, agreeing forms.

[7] Przepiórkowski (2007) distinguishes *syntactic* and *semantic* heads in syntactic annotation of a corpus of Polish.

[8] For technical reasons, the labels mark nodes rather than edges, representing both constituency and functional relations. The nodes refer to categorial information appropriate to words or phrases, as in the analysis of (6) above.

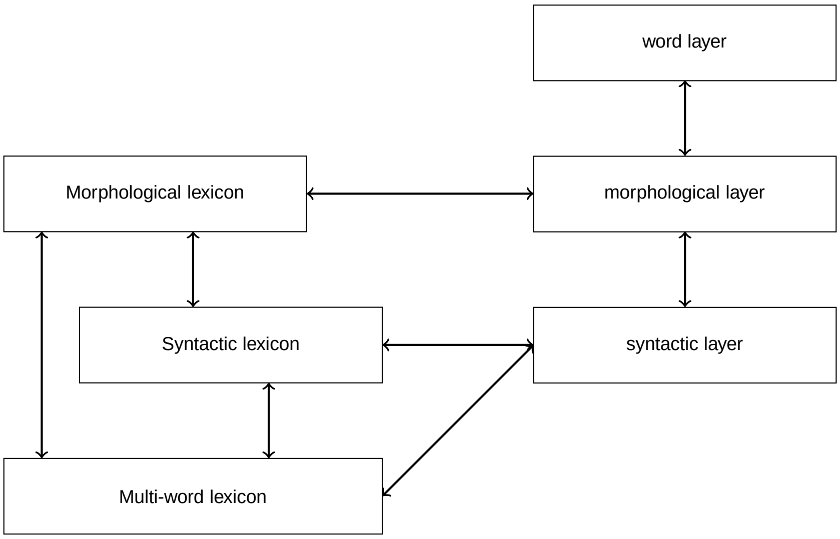


Figure 2: The setup of layers and lexical resources

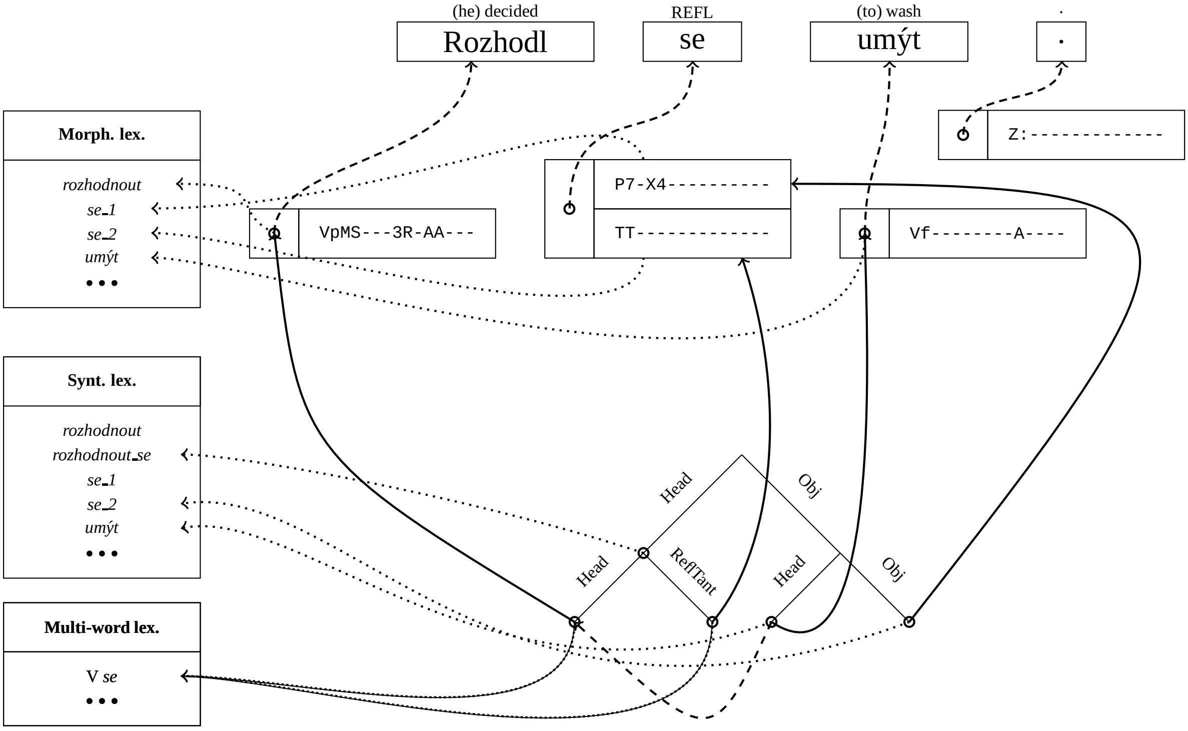


Figure 3: Solution to a case of haplology of a reflexive

whether in segmentation, morphology or syntax. Examples include valency slots with ambiguous case requirements filled by nouns exhibiting case syncretism as in (5) (Oliva, 2001), or structures involving PP-attachment ambiguity without a difference in meaning (4). Ambiguities of this type cannot be resolved even in a wide context.

(4) Uzavřeli mír s nepřítelem.
concluded peace with enemy
'They made peace with the enemy.'

(5) V továrně se využívá zařízení na výrobu
in factory REFL use device$_{nom/acc}$ for production
kyslíku.
oxygen
'In the plant a device for the production of oxygen is used.'

Unresolved ambiguity may also be preferable to an arbitrary decision in case of poor evidence.

The scheme accommodates inherently ambiguous or undecidable phenomena using underspecification and distribu-

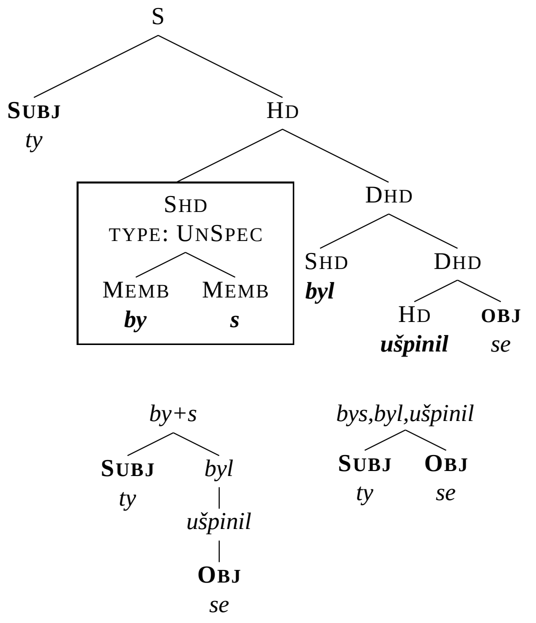


Figure 4: Three views of a single sentence

tive disjunction, both for category values and structures. Annotation of any kind can be missing; in the extreme case, syntactic structure of a sentence may consist of a mere list of words. A partial analysis may identify a word's head, its membership in a constituent, its syntactic function, or any combination of the above, while still leaving other syntactic relationships in the sentence unresolved.

To allow for such arbitrary underspecification, the skeleton structure is constituency-based, with a combination of binary and flat branching. Sub-constituents are specified by reference to a list of all constituents in sentence (6).[9]

(6) Zdravotnictví musí zachránit stát.
health service$_{nom/acc}$ must save state$_{nom/acc}$

#1 Health service must save the State.
#2 Health service must be saved by the government.

(7) Morphological analysis of (6) with some values unspecified:

$\boxed{1}$ zdravotnictví	*noun*, CASE=*X*, NUM=*sg*, GEND=*n*
$\boxed{2}$ musí	*verbfin*, PERS=*3*, NUM=*sg*
$\boxed{3}$ zachránit	*verbinf*
$\boxed{4}$ stát	*noun*, CASE=*Y*, NUM=*sg*, GEND=*m*

(8) Constituents in one of the two possible syntactic structures of (6), some boxed numbers refer to the forms above:

$\boxed{5}$ [$\boxed{3}$zachránit $\boxed{4}$stát]
$\boxed{6}$ [$\boxed{2}$musí $\boxed{5}$]
$\boxed{7}$ [$\boxed{1}$zdravotnictví $\boxed{6}$]

(9) Two possible structures with constraints on category values and overriding clauses:

#1 = $\boxed{7}$, *X=nom*, *Y=acc*

#2 = $\boxed{7}$, *X=acc*, *Y=nom*, $\boxed{1} \rightarrow \boxed{4}$, $\boxed{4} \rightarrow \boxed{1}$

While dependency structure requires a specification of heads and dependency links for all parts of the tree, constituency structure allows for leaving the status of a constituent and relations within an embedded constituent unspecified. Constituency structure may be more useful even for representing ambiguities, at least when they can be rendered as underspecifications. E.g., it is not clear what kind of structure is correct in appositive structures such as (10).

(10) vedoucí katedry profesor doktor Václav Novák
head department professor doctor Václav Novák
'Professor doctor Václav Novák, head of the department'

Ambiguities can either be present in the output of the parser, if it is run in an n-best mode, or they can be reconstructed by rules targeting typical cases. Moreover, PP-attachment ambiguities without semantic relevance are supposed to be tagged as such in the output of the parser, without generating multiple structures explicitly. For the time being, we intend to use the latter, somewhat unreliable, information wherever appropriate, and focus on experimenting with the reconstruction approach.

3. Converting dependency trees

Our syntactic trees are grown in a dependency-based nursery of McDonald's MST parser to the shape of the PDT a-level standard. The parser takes as its input a morphologically disambiguated sentence; the disambiguation is performed by a hybrid tagging system consisting of a rule-based component comprising about 2500 rules and a stochastic tagger. Syntactic trees produced by the parser are checked and rectified (see Section 4. below), and then converted to the internal annotation scheme and format, which differs from the input in the following aspects:

- In a different overall structure: the new scheme is based on constituency (phrase-structure) trees, e.g. with the subject a sister node to the clause's predicate.
- In a smaller set of syntactic functions (cf. Table 2).
- In a different account of word order, represented by links connecting unordered terminal nodes of the tree with their corresponding elements on the level of graphemics.
- In reference links, used, e.g., for connecting predicative elements (finite verb forms, infinitives, transgressives, nominal predicates) with their subject.

Rather than developing a full-fledged grammar of Czech from scratch and preparing appropriate training and testing data we used almost error-free training data in a format usable by the parser (customized for Czech) and also a vast amount of testing data in PDT (ca. 1.5 mil. tokens).

The conversion is performed by the application of a sequence of transforming rules to each input sentence. We demonstrate the conversion using (11) as an example.

(11) Most, který byl v havarijním stavu, by měl sloužit
bridge, which was in disrepair state, would have to-serve
dalších třicet let.
further thirty years

[9] Note that the example is not inherently ambiguous – it has two distinct interpretations, potentially distinguishable given an appropriate context or world knowledge.

> ‘The bridge, which was in a state of disrepair, should serve for a further thirty years.’

Sentence (11) is converted from the parser output (a-level of PDT) in Fig. 5 to the new format as in Fig. 6.

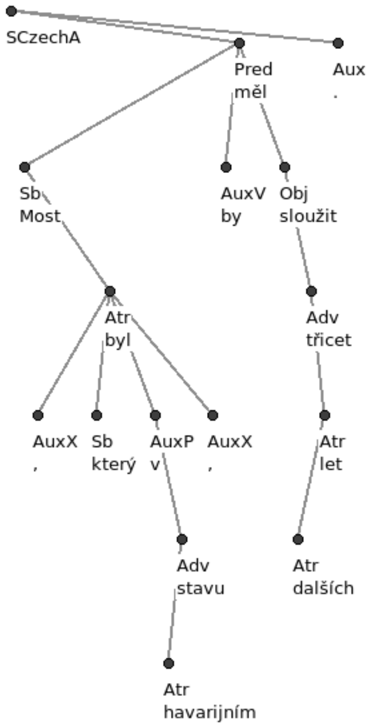


Figure 5: Syntactic tree in the PDT format

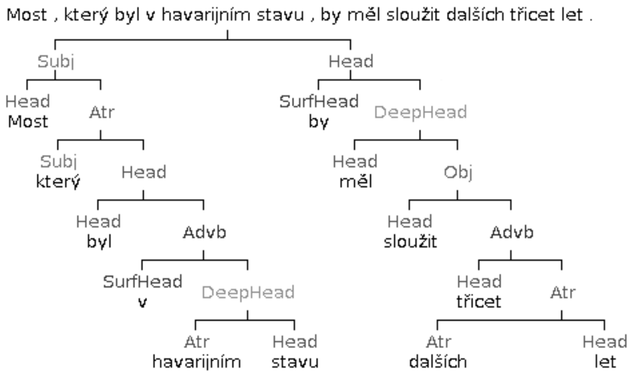


Figure 6: Syntactic tree in the new format

Some of the rules applied are merely technical or handle trivial operations on a single node. Other rules modify the geometry of the tree. Generally, the conversion of an input dependency tree to the corresponding constituency-based one proceeds recursively in a top-down direction, i.e., from the root of the input dependency tree down to the dependent nodes, and similarly, an output constituency-based tree is created. A rule is applied whenever the conversion algorithm finds the first appropriate bundle (= a simple tree).

In general, the input dependency bundle is converted in a following way: a new constituent is created having as its subconstituents a) all the constituents created by the transformation of the dependent nodes of the bundle and b) the constituent labeled by the output function HEAD or DEEPHEAD: this subconstituent will be assigned the word form, lemma and morphological properties of the head node of the dependency bundle.

A leaf of the input dependency tree is converted to the terminal constituent that is assigned the corresponding word form and its original function except for the nodes assigned the HEAD, DEEPHEAD or SURFHEAD function.

We shall briefly describe the main conversion rules, one of them a general rule, the others specific:

1. *General rule*. This rule converts a dependency bundle consistings of a head node and a non-empty set of its daughters to the corresponding constituent bundle. This bundle contains (i) the root and the nodes for the non-head daughters corresponding to the daughters of the input bundle and (ii) the node for the head daughter corresponding to the root of the input bundle: this node is assigned the HEAD function. The rule is applied to the input bundle whose head is a predicate and whose daughters are its non-subject complements.

2. *Subject rule and Auxiliary verb rule*. Both rules operate in a structurally identical way, the difference is in the syntactic functions. The *Subject rule* / *Auxiliary verb rule* take as input a bundle containing a root representing a verbal predicate and a *subject* / *auxiliary verb*, respectively, as one of the daughters of the root. On output, a binary constituent tree is created at the highest level: its head daughter corresponding to the verbal predicate is assigned the HEAD / DEEPHEAD function, the non-head daughter corresponds to the input *subject / auxiliary verb* and is assigned the SUBJ / SURFHEAD function. Furthermore, the head daughter is a root of another bundle: one of its daughters is assigned the HEAD function and the remaining non-head daughters correspond to the remaining sister nodes in the input.

3. *Prepositional group rule and Conjunction-clause rule*. These two rules also operate in a structurally identical way: they both transform an input dependency edge connecting a preposition / conjunction as the head element assigned the AuxP / AuxC function with the dependent node, referred to as X below. A binary tree is created on output: its root is assigned the function of the X node, the daughter corresponding to the preposition / conjunction is assigned the SURFHEAD function, the other one is assigned the DEEPHEAD function as well as the word form and morphological properties transferred from the input X node.

So far, 15 rules have been developed and tested on a subset of the PDT data. We plan to add new rules covering special phenomena encountered during intensive testing.

The conversion of the input structure of sentence (11) in Fig. 5 to the output structure in Fig. 6 can be briefly described as follows:

- First, the root of the input dependency tree labeled SCzechA as well as the node labeled AuxK representing a full-stop are left out, only the subtree with the root labeled by the Pred function will be converted.

- The Pred-labeled subtree will be processed in a left-to-right and top-down way, with every embedded bundle being recursively processed by the abovementioned rules. Thus, the Pred-labeled bundle with the three daughter nodes denoted Sb, AuxV and Obj will be processed first and subsequently the subtrees headed by them. The word forms and corresponding morphological tags (realized, in fact, by pointers to the morphological level and to lexicons, cf. Fig. 3 above) are recursively propagated down the constituency-based tree being created to be finally assigned to its leaf nodes.

In addition to the structure-changing rules used to generate phrase-structure trees complying with the new scheme, special rules adding reference links are applied.

Another group of rules, currently under development, are used to identify various substructures within the generated trees, such as:

- Agreement relations, such as subject – predicate, congruent attribute – noun, relative pronoun – antecedent

- Periphrastic verb forms including auxiliaries, such as conditionals, future and past tenses, passive

- Idioms and other specific types of collocations

- Inherently reflexive verbs or adjectives with the corresponding reflexive particles

- Surface/deep heads in structures of a specific type

- Non-projective (discontinuous) constructions (inferred from the surface order)

- Ambiguities undecidable even in a wider context (specific cases of PP-attachment and case syncretism)

Annotation of some of these structures (such as agreement relations and periphrastic forms) is not present in the treebank; the rules identifying them are invoked only after a user specifies his/her query to search for them in the treebank.

4. Automatic correction of results of the stochastic parser

Parsing unrestricted text by machine-learning techniques currently outperforms methods using hand-crafted rules, at least for Czech (Holan and Žabokrtský, 2006; Novák and Žabokrtský, 2007), therefore the input data for the conversion program are provided by stochastic parsers. Their accuracy up to now is, however, unsatisfactory: the best-performing parser available is the MST parser (McDonald et al., 2005) trained on PDT 2.5, whose success rate is 85.48% for unlabeled structures and 78.23% on labeled ones (measured on the *evaluation-test* subset of the PDT). A detailed analysis of the parser's results performed on the object corpus SYN2005 (100 mil. tokens of contemporary Czech) showed that it was possible to use linguistic knowledge for a reliable correction of many frequent parsing errors and thus to improve the overall accuracy of the parser. Therefore we developed our own rule-based correction tool with the set of rules being gradually enhanced.

During the development of this tool we annotated first the SYN2005 corpus by the stochastic parser. Then we manually checked the samples (ca 10,000 tokens altogether), in which we searched for recurrent errors, and subsequently we searched for errors in the whole corpus automatically. The most frequent errors were subject to further analysis and we tried to find reliable algorithms for error corrections. The parser makes both grave "grammatical" errors, where the resulting structure is totally inadmissible in the system of Czech syntax (e.g., two non-coordinated subjects depend on a single verb), and also common errors due to the parser's ignorance of valency, incorrect identification of head elements etc.

Automatic correction is performed by a modular software tool, employing extensive linguistic knowledge, represented, e.g., by various lists of lexical properties, including valency requirements. The tool identifies incorrect dependency structures, syntactic function labels and morphological tags. Whenever a well-known incorrect syntactic pattern is encountered, a corresponding correction routine is applied which chooses an appropriate correction according to the context and properties of the words in the erroneous structure. See (Jelínek, 2012) for more details.

One of the correction rules deals with subject incorrectly assigned to the noun in non-prepositional accusative. On the basis of context and properties of the incorrectly labeled word and its head verb the tool activates the appropriate linguistic rule which decides whether it should correct the syntactic function label (subject → object/adverbial), morphological tag (accusative → nominative in case of wrongly disambiguated case ambiguity) or the parent node in the dependency structure (some other appropriate head verb in the clause is selected).

So far we have created 30 correction rules, targeting more frequent errors for which reliable correction algorithms were found. One group of rules focuses on the basic dependency structure of a compound sentence consisting of verbs (they represent sentences in the PDT) and conjunctions. Another group of rules deals with incorrect syntactic functions and dependencies of syntactic nouns (e.g. the above-mentioned subject in accusative), the third group of rules rectifies the dependencies and functions of prepositional groups. The number of these rules grouped according their target domain and their success rate is shown in Table 3. The first column identifies the domain of the rules (dependency links and function labels of clauses, noun phrases, prepositional phrases and other rules), followed by the number of rules in the second column, the number of successful corrections of dependency links and syntactic function labels in the third and fourth columns, and the total number of successful corrections in the last column (some corrections modify dependency links as well as function labels). The success rate is calculated automatically on the *e-test* data set of PDT 2.5. The counts represent the number of successful corrections, decreased by the number of unsuccessful corrections, normalized per one million tokens. The last row shows the figures in percentages.

	rules	dependency	label	total
Clauses	6	1688	774	1744
NP	8	819	2066	2625
PP	11	834	7160	7722
Other	5	412	1390	1802
Total (ppm)		3753	11390	13893
Total (%)		0.38%	1.14%	1.39%

Table 3: Success rate of the correction modules

The resulting success rate improves from 85.48% unlabeled and 78.23% labeled accuracy to 85.86% unlabeled and 79.62% labeled accuracy. The correction tool can be further extended by additional rules.

In addition to extending the coverage of the correction tool, more procedures to improve the parsing results are in the pipeline, such as a rule-based pre-processing of both training and annotated data, a combination of multiple parsers,[10] a modification of the tagset to better serve the needs of parsing, and an extension of training data with text genres insufficiently covered in our original test set, such as fiction. We expect to raise the reliability of syntactic annotation above 87% unlabeled and 83% labeled by the end of the project.

5. Conclusion

We wish our treebank to match the size of POS-annotated corpora, while avoiding a theoretical bias by offering various views of syntactic annotation, based on a single core representation. The viability of this approach reflects the fact that linguistic theories share a broad common core. A sentence can then be visualized as a constituency-based or dependency-based structure with parameterizable underspecifications according to the user's wish. Three levels of representation (graphemic, morphological and syntactic) support the view of a bare input sentence and/or its morphological and syntactic annotation in various degrees of descriptive granularity. The system should satisfy demands of both an expert user and a student of syntax at higher elementary and secondary levels.

For a corpus of this size it would be unrealistic to count on manual checking of the output of automatic annotation tools. As a partial remedy, we use a rule-based correction module, targeting typical errors and inconsistencies. Together with visualisation options hiding very specific details or embedded structures, which a typical corpus user is expected to use as a preference, the effective error rate in the displayed data will be lower than in the output of the parser. We believe that the price for a significantly scaled-up treebank, paid in less reliable annotation, will be bearable for many tasks.

6. Acknowledgements

The authors are grateful to anonymous reviewers for very helpful comments. Work on this project was supported by the Grant Agency of the Czech Republic as project No. P406/10/0434.

7. References

Jan Hajič. 2006. Complex Corpus Annotation: The Prague Dependency Treebank. In Mária Šimková, editor, *Insight into the Slovak and Czech Corpus Linguistics*, pages 54–73, Bratislava, Slovakia. Veda.

Annette Hautli, Sebastian Sulger, and Miriam Butt. 2012. Adding an annotation layer to the hindi/urdu treebank. *Linguistic Issues in Language Technology – LiLT*, 7, January.

Tomáš Holan and Zdeněk Žabokrtský. 2006. Combining Czech dependency parsers. In Petr Sojka, Ivan Kopeček, and Karel Pala, editors, *TSD*, volume 4188 of *Lecture Notes in Computer Science*, pages 95–102. Springer.

Tomáš Jelínek. 2012. Automatic rule-based correction of stochastic syntactic annotation of Czech. In Markéta Ziková and Mojmír Dočekal, editors, *Slavic Languages in Formal Grammar. Proceedings of FDSL 8.5, Brno 2010*. Peter Lang, Frankfurt am Main.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 523–530, Stroudsburg, PA, USA. Association for Computational Linguistics.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC-2006*, pages 2216–2219, Genova. ELRA.

Václav Novák and Zdeněk Žabokrtský. 2007. Feature engineering in maximum spanning tree dependency parser. In Václav Matoušek and Pavel Mautner, editors, *Lecture Notes in Artificial Intelligence, Proceedings of the 10th International Conference on Text, Speech and Dialogue*, volume 4629 of *Lecture Notes in Computer Science*, pages 92–98. Springer, Berlin / Heidelberg.

Karel Oliva. 2001. On retaining ambiguity in disambiguated corpora. *TAL (Traitement Automatique des Langues)*, 42(2).

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

Adam Przepiórkowski and Alexandr Rosen. 2005. Czech and Polish raising/control with or without structure sharing. *Research in Language*, 3:33–66.

Adam Przepiórkowski. 2007. On heads and coordination in valence acquisition. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing (CICLing 2007)*, Lecture Notes in Computer Science, pages 50–61, Berlin. Springer-Verlag.

[10] We plan to test and tune at least one additional parser (Nivre et al., 2006).

Deep, Consistent and also Useful: Extracting Vistas from Deep Corpora for Shallower Tasks

João Silva, António Branco, Sérgio Castro, Francisco Costa

University of Lisbon
Edifício C6, Departamento de Informática,
Faculdade de Ciências, Universidade de Lisboa,
Campo Grande, 1749-016 Lisboa, Portugal
{jsilva, antonio.branco, sergio.castro, fcosta}@di.fc.ul.pt

Abstract

Annotated corpora are fundamental for NLP, and the trend in their development is to move towards datasets with increasingly detailed linguistic annotation. To cope with the complexity of producing such resources, some approaches rely on a supporting deep processing grammar that provides annotation that is rich and consistent over its morphological, syntactic and semantic layers. However, for some purposes, the deep linguistic corpora thus produced are "too deep" and unwieldy. For instance, if one wishes to obtain a probabilistic constituency parser by learning a model over a treebank, the full extent of the annotation created by a deep grammar is not needed and can even be detrimental to training. In this paper, we report on procedures that, starting from a deep dataset produced by a deep processing grammar, extract a variety of *vistas*—that is, subsets of the information contained in the full dataset. This allows taking a single base dataset as a starting point and, from it, deliver a variety of corpora that are more streamlined and focused on particular tasks.

1. Introduction

Annotated corpora are key resources for NLP. Not only are they important materials for researchers investigating linguistic phenomena, they also allow one to automatically obtain data-driven models of language and evaluate the tools thus produced.

Annotating corpora with human-verified linguistic information is a time-consuming and often error-prone task. Early treebanks for NLP, like the well-known Penn Treebank corpus, were built with the help of automatic annotation tools that were used to provide a preliminary annotation which was then manually corrected (Marcus et al., 1993).

Performing such corrections by hand can introduce formatting errors since manual changes may easily be malformed (e.g. misspelled categories, forgetting to close a bracket, etc). As such, the manual correction step is often aided by a tool that ensures that at least the linguistic information is well formed

However, as the linguistic information one wishes to include in the corpus grows in complexity, this approach becomes increasingly hard to adopt since the human annotator, even with the help of supporting software, has to keep track of too much interconnected information.

To address this issue, approaches to corpus annotation have come to rely on an auxiliary deep processing grammar as a way of producing rich annotation that is consistent over its morphological, syntactic and semantic layers. Two examples of such an approach are (Dipper, 2000), using an LFG framework, and (Oepen et al., 2002), under HPSG.

Despite these advantages in terms of consistency and depth of the information encoded, the annotation produced by such grammars is often too theory-specific or too unwieldy for certain purposes. For instance, if one wishes to train a probabilistic constituency parser, the linguistic information on grammatical functions and semantic roles present in the output of a deep grammar is not needed and, if integrated into the model, might actually be detrimental to the performance of the parser due to data-sparseness issues.

The image in Figure 1 helps to illustrate the problem. It shows the fully-fledged grammatical representation, under the HPSG framework, for the rather simple sentence *Todos os computadores têm um disco* (Eng.: All computers have a disk).[1]

Thus, it is desirable to have a process that allows extracting *vistas*—that is, subsets of the information contained in the full dataset—such as text annotated with part-of-speech tags, a plain constituency tree or a grammatical dependency graph.

In this paper we present a set of procedures that allow extracting several such vistas from a deep linguistic dataset. In particular, we will use a dataset that has been produced by a computational grammar based on the HPSG framework (Pollard and Sag, 1994; Sag and Wasow, 1999; Copestake, 2002).

Section 2 provides an overview of the grammar-supported annotation procedure used to produce the full deep dataset, while Section 3 describes the deep dataset itself. This is followed by Section 4, where the vista extraction procedures are presented. Section 5 provides an extrinsic evaluation of the extracted vistas by inducing probabilistic parsers over them. Finally, Section 6 concludes with final remarks.

2. Grammar-Supported Treebanking

A grammar-supported approach to corpus annotation consists in using a computational grammar to produce all possible analyses for a given sentence. What is then asked of the human annotator is to select the correct parse among all those that were returned. In such a setup, the task of the

[1] The printout is in a 6pt font. The arm and hand holding a pen are there just to give a sense of the size of the grammatical representation.

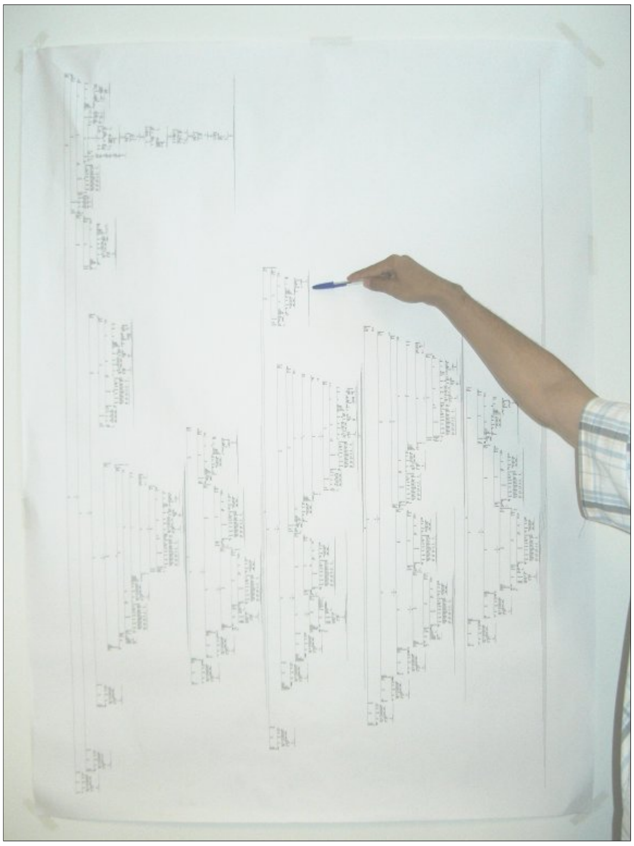


Figure 1: Full HPSG representation of the sentence

human annotator can be envisaged as being one of disambiguation.

Due to the inherent ambiguity of natural language, the parse forest that results from the grammar producing all possible analyses for a sentence may very well include hundreds of trees. Manually examining each individual tree in search for the correct one would prove unfeasible. Instead, the human disambiguator goes through a list of discriminants to reduce the number of parses. Discriminants are binary disambiguation decisions, many of which cut the parse forest in half.

For instance, PP-attachment is a common source of structural ambiguity, where a PP constituent may validly attach to more than one constituent in the parse tree. A discriminant would state whether the PP attaches to a given constituent. Choosing that discriminant as valid automatically prunes from the parse forest all parses where the attachment of that PP is different, while marking the discriminant as invalid discards all trees where the PP is attached to that same constituent.

For such an approach to work, it must be supported by a tool that provides the discriminants and handles the pruning of the parse forest in a manner that is unobtrusive for the human annotator. For datasets in the HPSG family, like the one used in this work, this can be done using the `[incr tsdb()]` tool (Oepen and Flickinger, 1998). Besides providing an interface for the disambiguation process described above, this tool integrates functionality for benchmarking, profiling and testing the grammar over test suites.

3. The Core Dataset

To create the core deep linguistic dataset from which the vistas will be extracted, we started with a corpus of Portuguese newspaper excerpts which had been previously annotated with manually verified shallow morpho-syntactic data, namely part-of-speech tags, lemmas, inflection features and information on named-entities.

This corpus was then treebanked according to the process outlined in Section 2. The supporting grammar that was used is LXGram, a deep computational grammar for Portuguese (Branco and Costa, 2008; Branco and Costa, 2010). It is worth of note that, for this dataset, annotation was done through a method of double-blind annotation followed by adjudication. In this setup, two human annotators work independently while pruning the parse forest returned by the grammar. If both annotators agree on the choice of an analysis, that analysis is added to the dataset. When the annotators disagree on what is the preferred analysis, a third human annotator, the adjudicator, is brought in to decide which analysis will be added to the dataset, if any (the adjudicator is free to choose a third analysis, rejecting the ones chosen by either annotator). This method of corpus annotation is resource-consuming, both in terms of human effort (three people are needed) and in terms of time (an adjudication round is required), but it allows a stricter quality control of the dataset being produced.

Due to the way it was built, the core dataset only contains those sentences that the supporting grammar was able to parse. It is formed by 5,422 sentences, most of which (4,644, or 86%) from newspaper text. The remaining sentences (778, or 14%) were part of the LXGram distribution and consist of sentences used for regression testing of the grammar.

4. Extracting Vistas

In this work we cover three vistas: the TreeBank, the DependencyBank and the PropBank, A TreeBank vista is a constituency tree, the familiar structure that represents the various constituents of the sentence and their level of aggregation. A DependencyBank vista, instead of giving a tree structure describing syntactic constituency, is a graph that relates pairs of words by a syntactic function (i.e. subject, direct object, modifier, etc). The PropBank is a dataset similar to the one described in (Kingsbury and Palmer, 2003) in that it consists of a layer of semantic role annotation that is added to phrases in the syntactic structure of the sentence. The format of these extended nodes in the PropBank tree is C-GF-SR, where C is the constituency tag, GF corresponds to the grammatical function and SR to the semantic role.

What is important to note regarding these three vistas is that the information contained in a PropBank is a super-set of the information present in the other two vistas. Our approach is then to take the PropBank as the main vista since the other two vistas, viz. the TreeBank and the DependencyBank, can in turn be obtained directly from it instead of having to extract each of them independently from the deep dataset.

For this we began by creating a PropBank extraction tool that runs over the deep representations resulting from the grammar-supported treebanking process. This tool makes use of the Tregex library created by the Stanford NLP Group (Levy and Andrew, 2006),[2] which provides a language for pattern matching over tree structures and regular expression matches over tree nodes.

4.1. PropBank Vista

The procedure that creates the PropBank consists of several steps, each having to deal with non-trivial issues. These steps are described in this Section.

4.1.1. Retrieving the Exported Tree

The deep representation of a sentence that is exported by `[incr tsdb()]` at the end of the manual disambiguation process includes the derivation tree, which encodes the rules that were used by the grammar during analysis of that sentence and the order in which they were applied. The exported deep representation also includes a second tree which has the same structure as the derivation tree, but where the rule names have been mapped into syntactic categories. This second tree, which we will call the *exported tree*, is taken by the tool as the starting point of the vista extraction procedure.

4.1.2. Tokenization

Due to the inner workings of `[incr tsdb()]`, the leafs in the exported tree are all converted to lowercase and truncated to the first 30 characters. Moreover, given the grammar used, the original newspaper corpus that was treebanked contains information not present in the deep dataset that has been created by the grammar (e.g. information on named entities). Thus, in order not to lose this data, we want it to be possible to incorporate it into the vistas. The most straightforward way of fixing each leaf is to replace it by the corresponding token from the original sentence.

For either of these procedures to work, leafs and tokens must be aligned. However, there is not a one-to-one correspondence between the leafs in the exported tree and the tokens in the sentence due to the original corpus and the grammar having different criteria for tokenization.

This is readily apparent in punctuation symbols, which are still attached to words in the exported tree, while they are found tokenized (i.e. detached from words) in the sentence. Given that the purpose of the tokenization stage is only to obtain a one-to-one correspondence between the leafs in the exported tree and the tokens in the sentence, punctuation symbols are simply detached from words and temporarily placed in a newly created sister node. The process of moving the punctuation symbols to their correct position in the final tree merits a slightly more detailed explanation and is addressed further ahead

4.1.3. Feature Bundles

Following the tokenization step, the leafs in the exported tree will be aligned one-to-one with the tokens from the original corpus, which allows the tool to easily copy the morpho-syntactic information from the corpus over to the tree as feature bundles that are appended to the leafs.

$$\underbrace{\text{comprei}}_{\text{word}} \,/\, \underbrace{\underbrace{\text{V}}_{\text{POS}} : \underbrace{\text{COMPRAR}}_{\text{lemma}} : \underbrace{\text{ppi-1s}}_{\text{inflection}} : \underbrace{\text{O}}_{\text{n.e.}}}_{\text{feature bundle}}$$

Figure 2: Leaf with added feature bundle

Figure 2 shows the breakdown of a feature bundle into its parts. Having the POS tag as a feature might at first seem unnecessary, since that information is given by the preterminal node of the tree, but the POS tagset of the original corpus is different from the one used by the grammar and, in this way, no information is lost. Named-entity information is encoded using a tagset like the one from the CoNLL shared task (Tjong Kim Sang, 2002): B is used for the first word in an entity, and I for any subsequent words in the same entity. A string representing the semantic type of the entity (e.g. PER for person, LOC for location, etc.) is appended. The letter O marks a word not belonging to any named entities.

Note that, in the following examples, the feature bundles appended to the leafs are not shown for the sake of readability.

4.1.4. Moving Punctuation

Punctuation symbols were detached from words during tokenization and placed in a temporary position. The current step is concerned with deciding where in the final tree to place the node with the detached punctuation symbol since its final position will depend on the syntactic construction the symbol is a part of.

Coordination is represented as a recursive tree structure where several constituents of the same type are combined together. Usually, a comma is used to separate each constituent, except for the last one which is delimited by an explicit conjunction, such as *e* (Eng.: and), *ou* (Eng.: or), etc.

As the example in Figure 3 shows, the comma is initially attached to the final word in a constituent of the enumeration. After being detached from the word, it is placed under a new node (PNT) which is in an adjunction position to the node to the right.

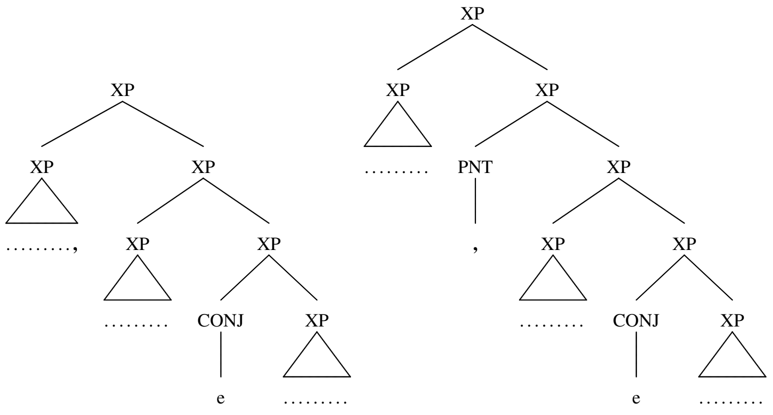
Appositions inside NPs are delimited by commas. This is made explicit in the tree representation by placing the apposition in a sub-tree that itself is delimited on either side by a pair of matching punctuation nodes, as shown in Figure 4. Parenthetical structures and quoted expressions are represented in a similar way. These are also the only situations where ternary nodes are used.

In all other cases, such as sentence-ending punctuation and topicalization, punctuation is adjoined far up as possible without crossing constituent boundaries. Figure 6 shows an example.

4.1.5. Collapsing Unary Chains

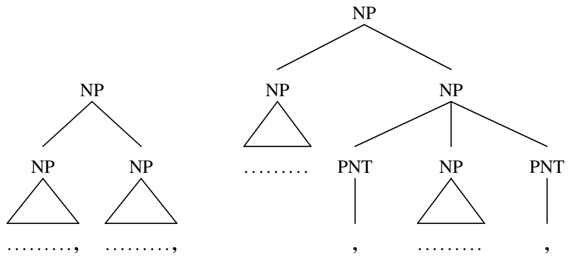
The syntactic representation of the exported tree contains unary chains of nodes with the same label. As mentioned above, this happens because the structure of the exported syntactic tree mirrors that of the derivation tree, which represents the rules applied by the grammar. Each node in these chains corresponds to the application of a unary morphological rule by the grammar (cf. (Copestake, 2002, Section 5.2) for more on such rules).

[2] Tregex website at `http://nlp.stanford.edu/software/tregex.shtml`.



(a) exported tree (b) final tree

Figure 3: Coordination



(a) exported tree (b) final tree

Figure 4: Apposition

For instance, the unary chain of three N nodes that dominates the word *computadores* (Eng.: computers) in Figure 5 corresponds, from the bottom up, to the application of the following morphological rules: COMPUTADOR (the rule for the lexical entry of the word), MASC-NOMINAL (flags a feature that marks the word as having gender inflection) and PL-NOMINAL (flags a feature that marks the word as having number inflection).

These various nodes in these unary chains are collapsed into a single node in the final tree.

4.1.6. Adding Phonetically Null Items

Nodes marking null subjects (*NULL*), null heads (*ELLIPSIS*), traces of constituents (*GAP*) and tough objects (*TOUGH*) are explicitly added to the final tree. There are several details concerning this step that are worth pointing out.

Pattern matching over the exported syntactic tree is not enough to always detect where one should add the nodes for phonetically null items. Instead, to do that, one must look at the derivation tree, since the relevant information can be found in the name of the derivation rule.

However, at this stage of processing, the syntactic tree and the derivation tree, which began by being isomorphic, do not have matching structures anymore, since the syntactic tree has been altered (viz. when moving punctuation and when collapsing unary chains). This issue was overcome by decorating the nodes in the syntactic tree with information taken from the derivation tree while both structures are still isomorphic.

Having decorated the syntactic tree, adding tree branches representing null subjects and null heads is quite straightforward.

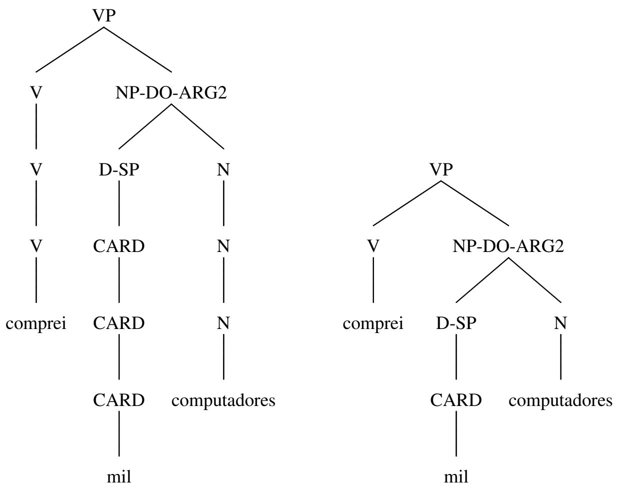
Null subjects are found by looking for SNS nodes in the exported syntactic tree, which are the way the grammar categorizes a sentence with a null subject. However, to properly assign a semantic role, the tool needs to look at the rule name from the corresponding node in the derivation tree, since the rule name indicates whether the missing NP-SJ node is an expletive (no semantic role), a passive construction (ARG2), a causative alternation (ARGA) or falls under the default case (ARG1).

Null heads are found by searching the derivation tree for certain rule names. The rule name not only indicates the category of the missing head (nominal or verbal) but also whether the head is the left or right child of the node.

Figure 6 shows an example of a parse tree with a null subject and a null nominal head.

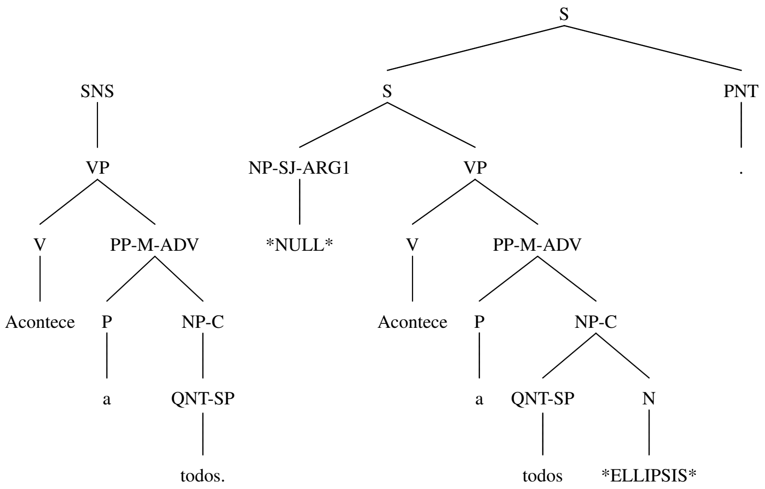
Nodes with a trace constituent are decorated by searching the derivation tree for a rule that indicates the extraction of a constituent and marking the corresponding node in the syntactic tree. The rule name also indicates whether the extracted constituent is on the left or on the right side of the node. The category of the extracted constituent is given by the usual HPSG slash notation, where a node labeled with X/Y indicates a constituent of type X that is missing a constituent of type Y.

When adding the trace, it suffices searching for the decorated node and add the *GAP* node as its left of right child, depending on the marking. In addition, the trace is



(a) exported tree (b) final tree

Figure 5: Unary chains
(Eng.: I bought a thousand computers)



(a) exported tree (b) final tree

Figure 6: Null subjects and null heads
(Eng.: Happens to everyone)

co-indexed with the displaced node by affixing the same index number to the trace and to the corresponding displaced constituent, as shown in Figure 7.

The displaced node is found by following the path of slashed constituents from the trace up to the topmost slash, which is the sister node of the displaced node.

When the sister of the topmost slash is not of the expected category it indicates a “tough” construction, and the trace node is marked with *TOUGH*, as shown in Figure 8.

4.1.7. Extending Semantic Role Annotation

The semantic role tags present in some of the nodes are at a different abstraction level than the constituency information conveyed by the phrase labels and tree structure. In particular, some role annotations show cross-tree dependency, where they need to refer to more than one constituent although the exported trees do not make this explicit.

This is the case with complex predicates, such as modals, auxiliaries and raising verbs. In such cases, the semantic role tag is suffixed with “cp” (for complex predicate). Anticausative constructions are handled in a similar way, but using “ac” as a suffix to the role tag.

For instance, the tree snippet shown in Figure 9 indicates that, though the NP is the subject of the VP, it is not the ARG1 of the head verb of the VP, but instead it is the ARG1 of some verb that is located down in the complex predicate topped by the VP.

Arguments of control verbs are handled in a similar manner, but one needs to look at the lexical type of the verb to determine whether it is a subject, direct object or indirect object control verb. To achieve this, the grammar lexicon is used to map the derivation rule for the lexical entry of a word (i.e. the pre-terminal node in the derivation tree) into the corresponding lexical type.

For instance, the ARG11 tag in Figure 10 indicates that the NP is both the subject of the control verb, *querem* (Eng.:

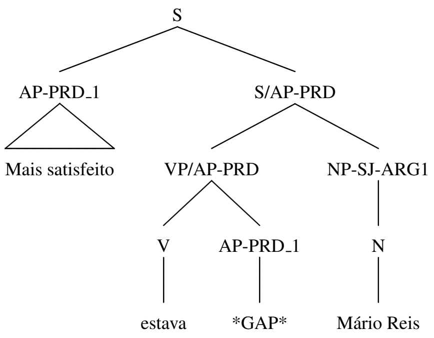


Figure 7: Traces and co-indexation
(Eng.: More pleased was Mário Reis)

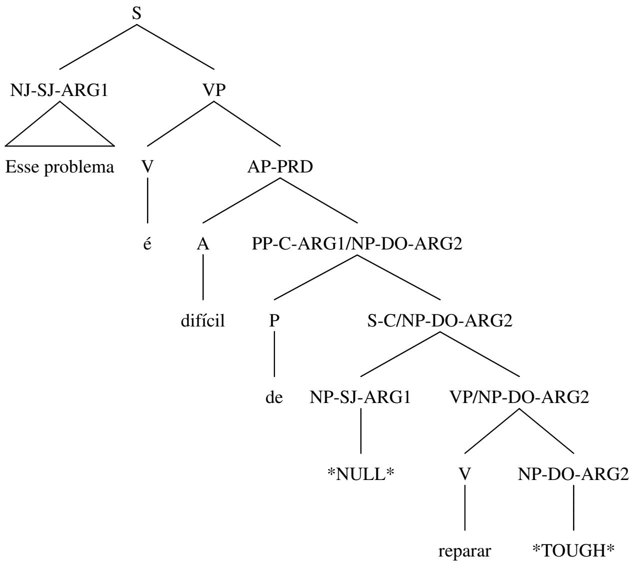


Figure 8: "Tough" constructions
(Eng.: That problem is tough to repair)

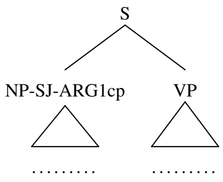


Figure 9: Complex predicate

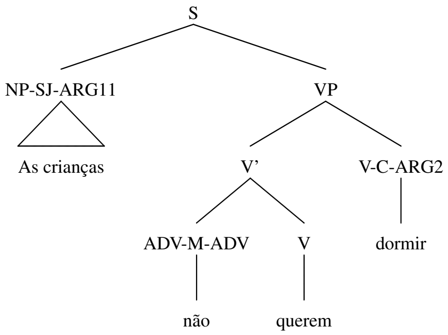


Figure 10: Control verbs
(Eng.: The children don't want to sleep)

want), and subject in the clause occurring as direct object of that verb.

4.2. TreeBank Vista

Having extracted the PropBank vista, the TreeBank vista is straightforward to obtain by simply discarding all information on grammatical function and semantic roles, leaving only the lexical and phrasal constituency information in the nodes of the tree.

4.3. DependencyBank Vista

To obtain the DependencyBank vista, one would like to make use of the extracted PropBank vista as an intermediate representation since it has already gone through an extensive normalization process. Fortunately, this is possible given that the trees that form the PropBank also include information on grammatical function in tags that are attached to the labels of some constituency nodes (e.g. SJ for subject, DO for direct object, M for modifier, etc). This gives us a straightforward way to automatically extract a dependency dataset from the PropBank.

Given that the PropBank adheres to an X-bar representation, phrasal nodes will have two children, one of which will be marked with a grammatical function. The (head of the) child that is marked is dependent on the (head of the) other child under the given grammatical function. The head of the phrasal node is the head of the unmarked child.

For instance, the tree fragment shown in Figure 11 yields a dependency where the head of ZP depends on the head of YP under relation F. The head of XP is the head of YP.

In the DependencyBank, displaced constituents are not represented by a *GAP* node. Instead, the head of the displaced node is dominated by the governor of its co-indexed node. For instance, in Figure 7 the head of the AP-PRD constituent is dependent on the verb *estava* (Eng.: was).

For complex predicates and anticausative constructions, the grammatical function tag is suffixed with the corresponding tag (i.e. either "cp" or "ac"). For instance, in Figure 9, the head of the NP is dependent on the head of the VP under the SJcp relation.

This procedure is carried out by a second tool that takes the PropBank as input and outputs the DependencyBank in the format of dependency triples and also in the widely-used CoNLL format (Nivre et al., 2007).

5. Evaluation

The extraction tool and the resulting vistas were evaluated extrinsically by measuring the performance of constituency and dependency probabilistic parsers trained over the corresponding vistas. The rationale for this approach being that a high quality dataset, with a consistent representation, should allow training probabilistic parsers that perform with high accuracy.

Note that, for the purpose of linguistic studies, both the TreeBank and the DependencyBank contain nodes that correspond to phonetically null items. These items, however, do not correspond to actual tokens that will appear in the input to the parser. Accordingly, they are removed from the TreeBank and DependencyBank vistas when training the parsers. In the TreeBank, the branches formed by a phonetically null item and the pre-terminal node immediately above it are pruned from the tree. Other information associated with these items, such as co-indexes and the slash notation used for traces, is also removed from the tree. In the PropBank, any dependencies involving the null items are discarded.

Since the focus is not on the development and tuning of the parsers, we opted for simply taking freely available third-party tools and running them out-of-the-box.

For constituency, we ran the Stanford parser (Klein and Manning, 2003), using the default parameters, over the 5,422 sentences in the TreeBank. This parser induces separate models, one for phrase-structure and one for lexical dependencies, which are then factored together during annotation. Following a standard 10-fold cross-validation approach, we obtained an 88% score under the Parseval metric. This is on par with the performance scores obtained by the same parser for English when training over the much larger Wall Street Journal dataset.

For evaluating the DependencyBank, we used MSTParser (McDonald et al., 2006), again using the default parameters. This parser works in two stages, the first assigning unlabeled dependency edges which are then labeled in the second stage using a sequence classifier. Under a 10-fold cross-validation evaluation methodology, we obtained a 87% labeled accuracy score, which is also a state-of-the-art score for this task.

6. Conclusion and Final Remarks

In this paper we presented and assessed a procedure for extracting vistas from a core deep dataset.

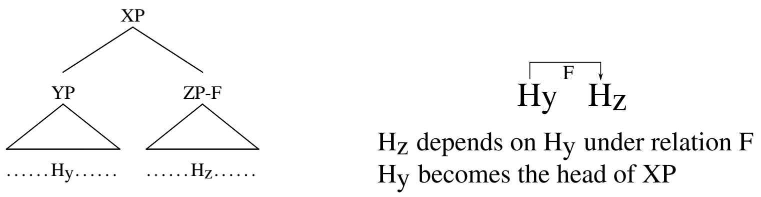
Deep processing grammars provide rich, accurate and consistent grammatical analyses for sentences, as well as much-needed support for the effective treebanking of corpora being annotated with rich linguistic information.

However, the output of such grammars may be too complex and unwieldy for what is required by certain tasks, motivating the need for creating procedures that extract streamlined and focused vistas. Such procedures allow taking a single, deep dataset as a starting point, with all the linguistic richness and annotation consistency that it offers, and extract subsets of the information contained in it.

For the work described in this paper, this core dataset is composed of 5,422 sentences of mostly newspaper text. It was created with the help of an HPSG deep processing grammar by manual double-blind disambiguation of the analyses produced by the grammar.

A tool was described that extracts a PropBank vista, a syntactic structure where phrases are enriched with a layer of semantic role annotation. The extracted PropBank was then used as a super-vista from which TreeBank and DependencyBank vistas were in turn also extracted.

These latter two vistas were evaluated by training probabilistic parsers over them, namely a constituency parser for the TreeBank and a dependency parser for the DependencyBank. In both cases, under 10-fold cross-validation, the parsers achieved state-of-the-art scores.



(a) PropBank snippet

(b) dependency relation

Figure 11: Extracting dependencies

7. References

António Branco and Francisco Costa. 2008. A computational grammar for deep linguistic processing of Portuguese: LX-Gram, version A.4.1. Technical Report DI-FCUL-TR-08-17, University of Lisbon.

António Branco and Francisco Costa. 2010. A deep linguistic processing grammar for Portuguese. In *Proceedings of the 9th Encontro para o Processamento Computacional da Língua Portuguesa Escrita e Falada (PROPOR)*, LNAI, pages 86–89. Springer.

Ann Copestake. 2002. *Implementing Typed Feature Structure Grammars*. CSLI Publications.

Stefanie Dipper. 2000. Grammar-based corpus annotation. In *Proceedings of the Workshop on Linguistically Interpreted Corpora*, pages 56–64.

Paul Kingsbury and Martha Palmer. 2003. Propbank: The next level of treebank. In *Proceedings of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 105–116.

Dan Klein and Christopher Manning. 2003. Fast exact inference with a factored model for NLP. *Advances in Neural Language Processing Systems*, 15:3–10.

Roger Levy and Galen Andrew. 2006. Tregex and Tsurgeon: Tools for querying and manipulating tree data structures. In *Proceedings of the 5th Language Resources and Evaluation Conference (LREC)*.

Mitchell Marcus, Mary Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the 10th Conference on Natural Language Learning (CoNLL)*, pages 216–220.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the 11th Conference on Natural Language Learning (CoNLL)*, pages 915–932.

Stephan Oepen and Daniel Flickinger. 1998. Towards systematic grammar profiling. Test suite technology ten years after. *Journal of Computer Speech and Language*, 12(4):411–436.

Stephan Oepen, Kristina Toutanova, Stuart Shieber, Christopher Manning, Dan Flickinger, and Thorsten Brants. 2002. The LinGO Redwoods treebank: Motivation and preliminary applications. In *Proceedings of the 19th Conference on Computational Linguistics (COLING)*.

Carl Pollard and Ivan Sag. 1994. *Head-Driven Phrase-Structure Grammar*. The University of Chicago Press.

Ivan Sag and Thomas Wasow. 1999. *Syntactic Theory: A Formal Introduction*. CSLI Publications.

Erik Tjong Kim Sang. 2002. Introduction to the CoNLL 2002 shared task: Language-independent named entity recognition. In *Proceedings of the 6th Conference on Natural Language Learning (CoNLL)*, pages 155–158.

Challenges in Converting between Treebanks: a Case Study from the HUTB

Rajesh Bhatt*[*] and Fei Xia[†]

[*]Univ. of Massachusetts
Amherst, MA 01003, USA
bhatt@linguist.umass.edu

[†]Univ. of Washington
Seattle, WA 98195, USA
fxia@uw.edu

Abstract

An important question for treebank development is whether high-quality conversion from one representation (e.g., dependency structure) to another representation (e.g., phrase structure) is possible, assuming that annotation guidelines exist for both representations. In this study, we demonstrate that the conversion is possible only under certain conditions, and even when the conditions are met, the conversion is complex as we need to examine the two sets of guidelines on a phenomenon-by-phenomenon basis and provide an intermediate representation for phenomena with incompatible analysis.

1. Introduction

There has been much interest in converting treebanks from one representation to another; for instance, from phrase structure to dependency structure or from phrase structure to other grammatical frameworks such as LTAG, HPSG, CCG, or LFG. While there have been many studies on converting between treebank representations (Collins et al., 1999; Xia and Palmer, 2001; Cahill et al., 2002; Nivre, 2003; Hockenmaier and Steedman, 2007), it is not clear how well the proposed conversion algorithms work because, for the treebanks used in those studies, annotation guidelines are available only for one of the two representations.

Compared to other existing treebanks, the Hindi/Urdu Treebank (HUTB) (Palmer et al., 2009) is unusual in that it contains three layers: dependency structure (DS), PropBank-style annotation (PB) (Kingsbury et al., 2002) for predicate-argument structure, and an independently motivated phrase-structure (PS) annotation which is automatically derived from the DS plus the PB. Because the treebank has detailed guidelines for all three layers and hundreds of guideline sentences with all three layers manually annotated, the treebank is a good resource for evaluating the performance of conversion algorithms. More importantly, the DS guidelines and the PS guidelines are based on different linguistic theories and the DS and the PS, as two representations, have different properties. While the idea of automatically creating PS trees from the DS and PB is appealing as it reduces the amount of human annotation, it raises many interesting questions:

- Does a *general-purpose*, high-quality DS-to-PS conversion algorithm exist? That is, an algorithm that performs well for any given sets of DS and PS guidelines?
- How much "freedom" do the designers of the DS and PS guidelines have in choosing analyses for linguistic phenomena?
- What kind of information should be included in the DS and PB in order to make the automatic conversion possible?

These questions are difficult to answer in the abstract. In this paper, we discuss them in the context of our experiences with the construction of the multi-representational Hindi-Urdu Treebank which involves automatic generation of the PS from the DS and PB.

2. An Overview of the HUTB

The HUTB (Palmer et al., 2009) has been developed by our colleagues and us since 2008. It has three layers, as explained below.

2.1. Dependency Structure (DS)

The HUTB chose the Paninian grammatical model (Bharati et al., 1995; Begum et al., 2008) as the basis of the DS analysis. The sentence is treated as a series of modifier-modified relations which has a primary modified (generally the main verb). The relations are of two types: karaka (roles of various participants in an action, i.e., arguments, notated as k1-k6) and others (roles such as purpose and location, i.e. adjuncts).

2.2. Propbank (PB)

PropBanking is a semantic layer of annotation that adds predicate argument structures to syntactic representations (Palmer et al., 2005). For each verb, PropBank represents the information about the arguments that appear with the verb in its corresponding frame file. The arguments of the verbs are labeled using a small set of numbered arguments, e.g. Arg0, Arg1, Arg2, etc. Additionally, verb modifiers are annotated using functional tags such as ArgM-LOC, ArgM-TMP, ArgM-MNR.

2.3. Phrase Structure (PS)

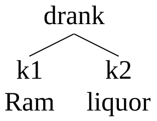
The PS guidelines are inspired by the Principles-and-Parameters methodology of Chomsky (1981). PS assumes a binary branching representation, where a minimal clause distinguishes at most two positions structurally (the core arguments). Displacement of core arguments from their canonical positions is represented via traces.

2.4. Overall Process

The treebank has three sets of annotation guidelines, one for each layer. The treebank is created in three steps. The first step is the manual annotation of DS. The second step is PropBanking, which focuses on adding the lexical predicate-argument structure on the top of DS. The third step is the automatic creation of PS, which is done by a DS-to-PS conversion process that takes DS and PropBank as input and generates PS as output. Figure 1 shows the three layers for a simple sentence. For the sake of saving space and readability for non-Hindi speakers, Hindi sentences in this paper are written as English words in Hindi word order.

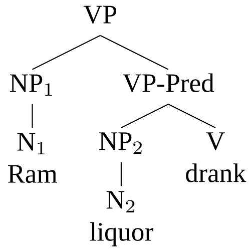
(1) a. Ram liquor drank ('Ram drank liquor')

b. DS tree:



c. PB annotation:
Predicate: drank
Frame id: drink.1
Arg0: Ram
Arg1: liquor

d. PS tree:



3. DS-to-PS Conversion

While the input to the process includes DS and PB, for the sake of simplicity, in the rest of the paper we will simply call the process *DS-to-PS conversion*, with the understanding that the *DS* in this context also includes information from the PB.

3.1. Previous work on DS-to-PS conversion

The common setting of a DS-to-PS conversion process is given in Figure 1, which has three stages. In the training stage, the input is a set of (DS, PS) pairs, $\{(DS_i, PS_i)\}$; the output is a model, a set of conversion rules, or something else depending on the conversion algorithm. In the test stage, a DS tree, DS_t, is sent to the test module, along with the output of the training stage; the test module produces a PS tree, $PS_t^{(0)}$. In the evaluation stage, the output of the test stage is compared with the gold standard, PS_t, and some scores (e.g., labeled F-score) are produced as a measure for the overall performance of the conversion algorithm.

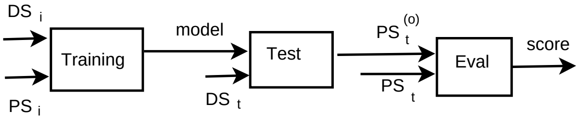


Figure 1: DS-to-PS conversion and evaluation

The previous DS-to-PS conversion algorithms can be divided into two types depending on whether there is an explicit training stage. In (Collins et al., 1999; Xia and Palmer, 2001), the conversion algorithms were purely rule-based: the rules were created by hand and used to build $PS_t^{(0)}$ given DS_t; there was no training stage. Xia et al. (2009) automated the conversion process by introducing the concept of *consistency* between a DS and a PS and proposing a process that extracts conversion rules from consistent (DS, PS) pairs in the training stage; in the test stage, the extracted rules were applied to an input DS_t to generate $PS_t^{(0)}$. One limitation of these previous studies is that they evaluated their conversion algorithms on treebanks for which annotation guidelines and manual annotation exist only for one of the two representations, and, therefore, it is not clear how well the algorithms truly performed.

Bhatt et al. (2011) proposed an analytical framework for determining how difficult it would be to convert one representation to another representation (DS and PS in this case) when each representation has its own annotation guidelines. They demonstrated that the conversion procedure must examine guidelines on a phenomenon-by-phenomenon basis, and for each phenomenon, there are three possible scenarios: (1) the two guidelines have *compatible* analyses; (2) they have incompatible analyses; and (3) one represents the phenomenon but the other does not. In the first case, automatic conversion is fairly direct; in the second case, one needs to study the DS and PS analyses for the phenomenon and provide an intermediate representation to bring the gap; in the third case, additional information is required to achieve the conversion.

Bhatt et al. (2011) defined *compatibility* of analyses based on *consistency* of (DS,PS) pairs. As defined in (Xia et al., 2009), a PS and a DS are called *consistent* if and only if there exists an assignment of head words for the internal nodes in PS such that after the flatten operation and the label replacement operation, the new PS is identical to the DS once we ignore the dependency types in the DS.[1] For instance, the DS and the PS in Ex (1) are consistent because when we choose *V* as the head child of *VP-Pred*, and *VP-Pred* as the head child of *VP*, we will merge these three nodes in the flatten operation and relabel the merged node with the head word *drank*; similarly, N_1 and NP_1 are merged and relabeled as *Ram*, and N_2 and NP_2 are merged and relabeled as *liquor*. The resulting tree is identical to the DS tree if we ignore the dependency types.

Given a linguistic phenomenon, let *D* be the set of (DS, PS) pairs for the sentences in the guidelines for that phenomenon. The analyses in the DS and PS guidelines are called *compatible* if and only if every (DS, PS) pair in *D* is consistent.

3.2. Our conversion process

Before we get into the details of our conversion process, it is important to address the first question raised in Section 1.: does a general-purpose DS-to-PS conversion algorithm exist that works well for any given sets of DS and PS guidelines? Referring to the flowchart in Figure 1, a conversion algorithm would correspond to the training and test modules; the DS and PS guidelines would correspond to $(DS_{i/k}, PS_{i/k})$ pairs; a *general-purpose, high-quality* algorithm would be one that produces $PS_t^{(0)}$ that is very similar to PS_t and therefore leads to a high evaluation score, no matter what $(DS_{i/k}, PS_{i/k})$ pairs look like.

Note that the flowchart shows the same setting as any machine learning (ML) system if we just replace *DS* with the input of a ML task (e.g., *sentence* for the parsing task) and replace *PS* with the output of an ML task (e.g., *parse tree* for the parsing task); therefore, in theory, it is possible that one can build a general-purpose, high-quality conversion algorithm, just like one can build a good statistical parser. On the other hand, while it is likely that a small number of (DS, PS) pairs exist for the language of our interest (e.g., trees for sentences in the annotation guidelines), we cannot assume that the number of pairs would be very large (say tens or hundreds of thousand pairs) because if there are so many (DS, PS) pairs available, DS-to-PS conversion is no longer important as one can easily create a PS treebank from these pairs. Just like there does not exist a general-purpose parser that performs well when trained on a few hundreds of (sentence, parse tree) pairs, we doubt that there exists a general-purpose DS-to-PS conversion algorithm that performs well when *trained* on a few hundreds (DS, PS) pairs, because, in the worst scenario, the analyses chosen by the DS and PS guidelines for linguistic phenomena can be so different that building PS from a given DS is not much easier than building a parse tree from a sentence.

Instead, we believe that high-quality DS-to-PS conversion is possible only if all of the following conditions hold: (1) the analyses chosen by DS and PS guidelines for most linguistic phenomena are compatible; (2) for the phenomena with incompatible analyses, the incompatibility can be resolved by simple transformations; and (3) for phenomena that are represented in the PS but not in the DS, the additional information needed to build PS is available from the PB or other sources.

If these conditions hold, high-quality DS-to-PS conversion is possible. Our conversion process for creating a PS from DS plus PB is illustrated in Figure 2.[2] It has two main modules: the first module handles phenomena with incompatible DS/PS analyses or phenomena represented only in the PS analyses. The input are DS and PB, and the output is a new, "extended" dependency structure called *DS+*. *DS+* should be consistent with the desired PS according to the PS guidelines.

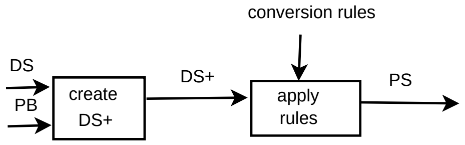


Figure 2: Building PS from DS and PB

The second module creates the PS from *DS+* by applying *conversion rules*. The *conversion rule* is a (DS-pattern, PS-pattern) pair, which says the DS pattern in a DS would correspond to the PS pattern in a PS tree. Figure 3 shows two conversion rules that will be used to create the PS in (1d) from the DS in (1b). The first rule says that when a verb in a DS has a left *k1* dependent whose head is a noun, the corresponding PS will have a *VP* node, which has an *NP* child followed by a *VP-Pred* child. The second rule is interpreted similarly. The conversion rules can be created by hand or extracted from consistent (DS, PS) pairs. The formal definition of conversion rules and the algorithms for extracting rules from (DS,PS) pairs and applying rules to generate a PS were discussed in (Xia et al., 2009).

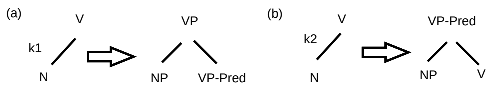


Figure 3: Two conversion rules

4. Handling incompatibility

As discussed in the previous section, we propose to use DS+ to handle phenomena with incompatible DS and PS analyses. The question is what DS+ should look like and how it can be created from the input DS and PB. In order to answer the question, we first need to understand the main sources of incompatibility. We will then go over seven linguistic phenomena that have

[1] The *flatten* operation merges all the internal nodes in the PS with their head child; the *label replacement* operation replaces the label of an internal node with its head word.

[2] This corresponds to the test stage in Figure 1.

incompatible analyses in the HUTB and show the corresponding DS+.

4.1. Main sources of incompatibility

Given that DS and PS guidelines are often based on different linguistic theories, there can be many reasons for incompatibility between the DS and PS analyses. Some instances of incompatibility could be accidental in that the DS and the PS might just choose distinct analyses even though in principle they could have picked the same analysis. Since we have developed the DS and PS guidelines in tandem, we have attempted to minimize the accidental incompatibilities. That leaves us with the more deep-seated sources of incompatibility. Here, we discuss three main reasons that cover the majority of such incompatibility in the HUTB.

The first reason is that one side chooses to represent certain relationships or distinctions, but the other does not. One example is the unaccusative vs. unergative distinction, which is represented in PS, not in DS. In order to create the desired PS, the list of unaccusative verbs has to be available from other sources, and in the HUTB that information comes from the PB.

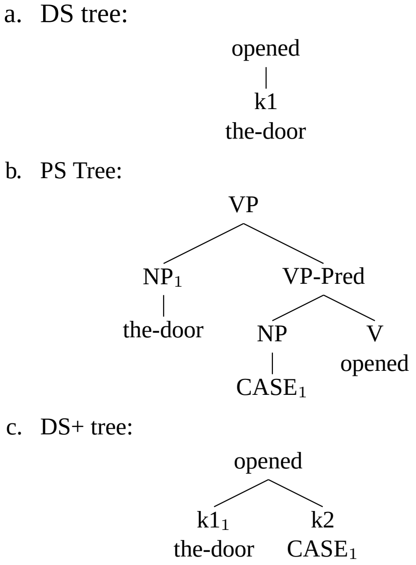
The second reason is due to different representational vehicles that are available in DS and PS. In the HUTB, DS represents information through structural means, dependency labels (e.g., k1 and k2), or attributes in the nodes. PS represents information through structural means, syntactic labels (e.g., *NP*), and coindexation (e.g., between a trace and its antecedent). Consequently, the DS and the PS could represent the same information, but through different vehicles. The corresponding DS and PS trees could end up being inconsistent, because the definitions of consistency and compatibility look at tree structure only. In the HUTB, the analyses for passive, small clause, support verb, and causative fall into this category.

The third reason is due to the differences in handling word order by the DS and the PS.[3] The DS in the HUTB allows for non-projective trees and it does not have a notion of canonical word order. In contrast, the PS tree in the HUTB must be projective and it assumes that the core arguments are generated in distinguished structural positions which implies that there is an inherent notion of canonical word order. Consequently any DS trees that are non-projective or in which core arguments appear in non-canonical word order would be inconsistent with the corresponding PS trees.

4.2. Unaccusatives vs. Unergatives

In the HUTB, the DS treats all intransitives alike while the PS makes a structural distinction between unergatives and unaccusatives: the PS treats the subject of an unaccusative as originating in the object position, as indicated by the empty category *CASE* and the coindexation between the subject and the object positions; the PS treats the subject of an unergative as originating in the subject position and there is no movement involved. Two examples are given in Ex (2) and (3).

(2) unaccusative: The door opened.

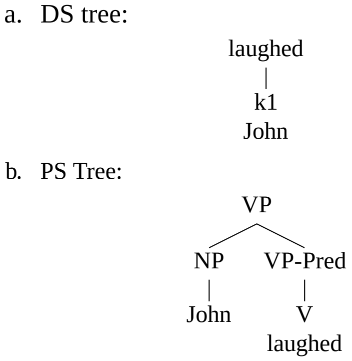


a. DS tree:

b. PS Tree:

c. DS+ tree:

(3) Unergative: John laughed.



a. DS tree:

b. PS Tree:

For automatic conversion to be a possibility, information about whether a given verb is unergative or unaccusative needs to be available. In the HUTB, that information is provided in the PB. The next question is what DS+ looks like. One intuition is that DS+ should include all the empty categories (ECs) appearing in the PS. In this case, we need to insert *CASE* to the DS+. Based on the DS and PS trees in Ex (2), *V* is the head child of *VP-Pred*; therefore, *CASE* should depend on *opened* in the DS+. As for its dependency type, *CASE* is in the canonical object position in the PS, and the dependency type for that position is *k2* in general, as shown in the second rule in Figure 3. Therefore, we will insert *CASE* as a dependent of *opened* with the type *k2*, and we use coindexation to link the EC and its antecedent. The resulting tree is in Ex (2c). In contrast, unergatives do not require DS+ (that is, its DS+ is the same as DS).

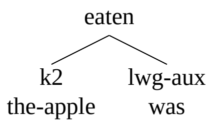
[3] This can be seen as a special case of the first reason; that is, the PS represents word order, whereas DS does not. But because word order is so salient and common, we treat this as a separate case.

4.3. Passive

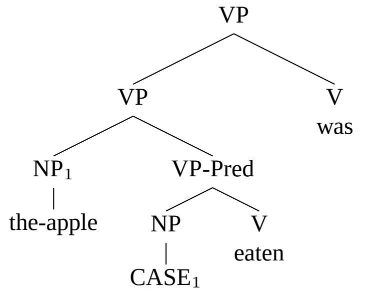
In the HUTB, both DS and PS indicate that the subject of the passive is related to the object position: the DS uses dependency type *k2*, and the PS uses the EC *CASE* and the coindexation between the subject and the object positions, as shown in Ex (4).[4]

(4) The apple eaten was ('The apple was eaten')

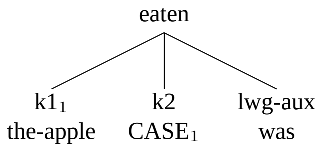
a. DS tree:



b. PS Tree:



c. DS+ Tree:



To detect passive is easy because a passive verb in the DS has a feature *passive='+'*. The DS+ for passive is similar to the one for unaccusative except that we change the dependency type of *the apple* from *k2* to *k1* because the phrase is in the canonical subject position in the PS, not the canonical object position.

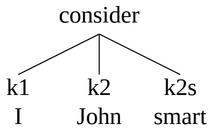
4.4. Small clause and support verb

The two phenomena we have discussed so far involve only one predicate (the unaccusative verb or the passivized verb) in both DS and PS. Small clauses are different in that they involve two predicates, as shown in Ex (5): *consider* and *smart*. *John* is related to both predicates: it gets case from *consider* and semantically it is an argument of *smart*.

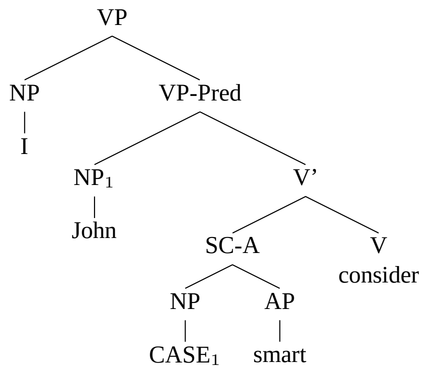
Both the DS and the PS represent these relations, but they do so in different ways. The DS represents the first relation by making *John* a dependent of *consider*; it represents the second relation by using the dependency type *k2s* for *smart*, and *k2s* encodes the information that its semantic argument has the label *k2*. The PS represents the two relations by inserting an EC *CASE* and coindexing it with *John*, and thus represents both relations structurally. Creating DS+ is simple; we just need to insert an EC *CASE* as a *k1* dependent of *smart* and coindex it with *John*.

(5) I John smart consider ('I consider John smart')

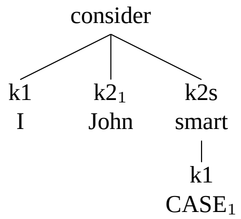
a. DS tree:



b. PS tree:



c. DS+ tree:



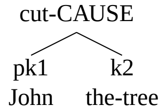
In the support verb construction, a verb and a noun form a complex predicate. An example is *John bicycle theft did* ('John stole a bicyle'), where *did* and *theft* form a complex predicate. Our treatment of support verb is similar to that of small clauses.

4.5. Causatives

Causative is another example where DS and PS represent the same information through different means. An example is given in Ex (6). The DS analyzes the causativized verb as a single head, but it labels the causer *John* as *pk1* (not *k1*), indicating that *John* is not really an argument of *cut*, but an argument of the causative part of *cut-CAUSE*. The PS represents the causativized verb as two independent heads: an EC, *CAUSE*, as the head of the higher clause, and the original verb as the head of the lower clause. In addition, the PS indicates the implicit intermediate agent explicitly as an EC, *IMP-ARG* (implicit argument).

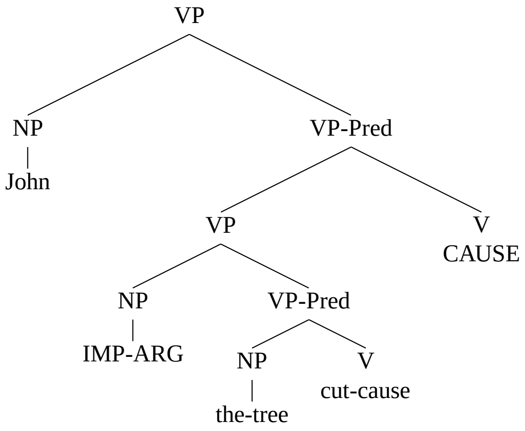
(6) John the-tree cut-CAUSE ('John caused the tree to be cut'.)

a. DS tree:

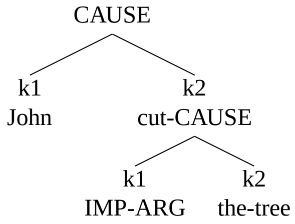


b. PS tree:

[4] The dependency type *lwg-aux* indicates that *was* is an auxiliary verb, and the word and its head *eaten* form a local word group (*lwg*)



c. DS+ tree:



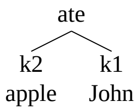
To create DS+, we insert two ECs: *CAUSE*, as the head of the higher clause, and another EC, *IMP-ARG* as a *k1* dependent of the lower clause. Furthermore, the causee becomes a dependent of *CAUSE* and its label is changed from *pk1* to *k1*.

4.6. Movement

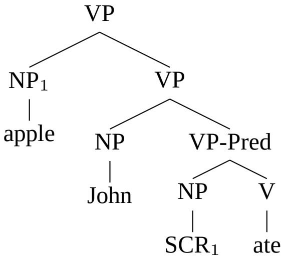
The last type of divergences involve the treatment of movement. In HUTB, the DS is not concerned about non-canonical word order; sentences with different word orders will have the same DS if we treat the DS as an unordered tree. In contrast, the PS assumes that the dependents of a head have a canonical order and if they are not in the canonical order, that is due to syntactic movement which is represented by an EC in the base position and a coindex between the base position and the surface position. An example is in Ex (7).

(7) apple John ate ('John ate an apple')

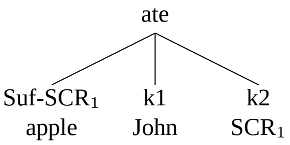
a. DS tree:



b. PS tree:



c. DS+

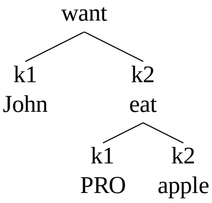


To create DS+ for this example, we need to know the canonical order of arguments of a verb. In Hindi, the order is *k1, k4, k2, verb*. By checking the word order in the sentence, we can detect the predicates whose dependents are not in the canonical order. We then use simple heuristics to determine which dependent is *moved* (in this case, it is *k2*). Next, we insert an EC *SCR* to DS+ as a *k2* dependent of the verb, replace the label of *apple* from *k2* to a new dependency type *Suf-SCR* (for the surface position of a scrambled element), and coindex the EC with *apple*.

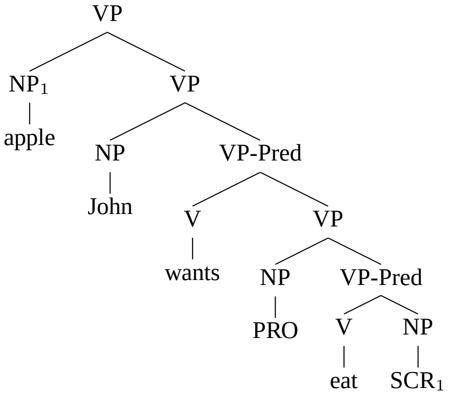
Movement in Ex (7) does not cause non-projectivity, because it does not cross the boundary of the clause. When movement crosses a clause boundary, its DS tree will be non-projective, see Ex (8).

(8) apple, John eat want ('John wants to eat an apple')

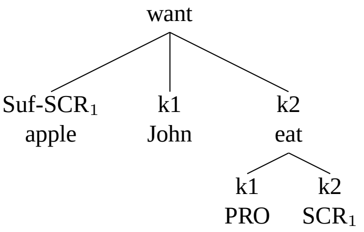
a. DS tree:[5]



b. PS tree:



c. DS+ tree:



Detecting non-projectivity is trivial given the original sentence and the DS. The creation of DS+ is similar to the process for the local movement, except that the

[5] The EC, PRO, is actually added by the PB.

moved element (*apple* in this example) will be moved up along the path from its parent to the root of the DS until its new position resolves the non-projectivity. In this example, *apple* becomes a child of *want* in DS+. Its dependency relation to *eat* is implicit as its trace *SCR* depends on *eat*.

Most movement in Hindi is to the left, as in Ex (7) and (8). But movement to the right is possible. The creation of DS+ for rightward movement is not discussed here due to the limitation of space.

4.7. Combinations

So far we have discussed seven phenomena where DS+ is needed to bridge the differences between DS and PS analyses. For each phenomenon, we have created a *rule* (i.e., a piece of code) that detects the phenomenon in a given DS plus PB and builds the DS+ accordingly.

Some of these phenomena can co-occur to the same predicate and its dependents in a DS; for instance, the arguments of a causative verb can undergo leftward or rightward movement ('a book John caused Mary to be given'); the main verb in a small clause construction can be passivized (e.g., 'John is considered intelligent'). We call them *combinations* of phenomena. The question is whether we can handle such combinations without writing more rules. In other words, (1) what combinations of phenomena are possible? (2) For these combinations, can the correct DS+ be created by applying the rules for individual phenomena in a certain order? (3) If so, what should the order be?

For the first question, some combinations are impossible. For instance, an unaccusative verb (e.g., *break* in 'window broke') lacks an external argument and cannot be passivized, so unaccusative + passive does not exist. Based on our observations on grammaticality of various combinations, we group the seven phenomena into five groups so that all the possible combinations consist of at most one phenomenon from each group:

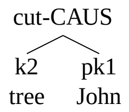
- Group 1: unaccusative, passive
- Group 2: small clause, support verb
- Group 3: causative
- Group 4: rightward movement
- Group 5: leftward movement

We show that the answer to the second question is *yes* by using the ordering based on the five groups; that is, applying the two rules in Group 1 first, followed by the rules for Group 2, 3, 4, and 5. This is the same order if we sort the rules based on the size of the region affected by the rules: Rules in Group 1 only affect a simple clause; rules in Group 2 affect a clause that contains a small clause; the rule in Group 3 affects a higher clause and a lower clause; the rules in Groups 4 and 5 can affect multiple clauses as movement can cross clause boundaries.[6]

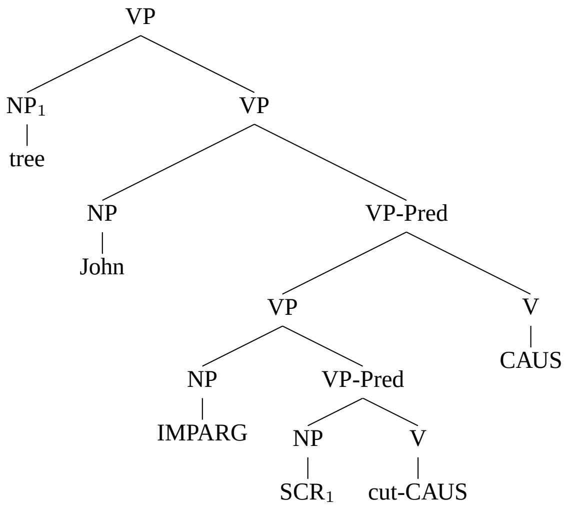
Now that we have fixed the ordering of the rules, we test whether applying rules in that order would produce the desired DS+. It turns out that the answer is indeed affirmative. Due to the limitation of space, we will just show an example. In (9), (b) and (c) are the input DS and the desired PS, respectively; (d) is the resulting DS+ after applying the rule for causative to (b); (e) is the resulting DS+ after applying the rule for leftward movement to (d), and it is indeed the DS+ that we want to create and it is consistent with the PS.

(9) a. tree John cut-CAUS ('John caused the tree to be cut')

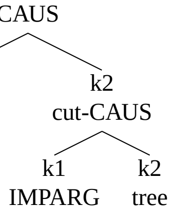
b. DS:



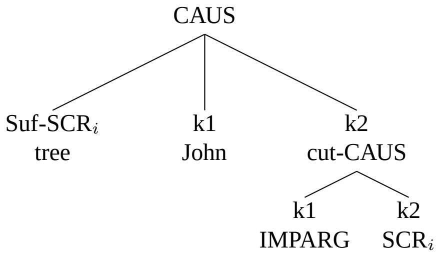
c. PS:



d. DS+ after applying the causative rule to (b):



e. DS+ after applying the leftward movement rule to (d):



[6] Note that not all the orderings would yield the desired PS. For instance, leftward movement (group 5) should be handled after rightward movement (group 4) because in the DS+ for rightward movement, a trace for the moved element needs to be immediately before the verb, which might not be the canonical position for that element and this can trigger leftward movement.

5. Discussion

The previous section went over seven phenomena for which DS and PS analyses are incompatible. Due to the limit of space, we used very simple examples and did not explain all the details. These details mean that the step of creating DS+ can be very complex; it requires manually going through all the phenomena with incompatible DS and PS analyses, and for each phenomenon determining what are the diagnostic tests for detecting the phenomenon and what DS+ should look like, and then writing rules to build the DS+. Furthermore, one needs to check whether or not the combinations of phenomena can be handled by applying rules in a particular order. Now we are ready to address the questions raised in Section 1.

First, a general-purpose, high-quality DS-to-PS conversion algorithm is unlikely to exist, because the number of (DS, PS) pairs is too small for building a statistical system; consequently, any high-quality conversion would require manual comparison of DS and PS analyses for each linguistic phenomenon; this process is time consuming and cannot be fully automated.

Second, the DS and PS guideline designers have some freedom in choosing analyses for linguistic phenomena, because DS+ serves as a vehicle to bridge the gap between the DS and PS analyses. However, the bigger the gap is, the more complex the module for creating DS+ will be. Therefore, when DS and PS guideline designers choose incompatible analyses, the decisions should be well-motivated.

Third, as shown in Figure 2, the input to the conversion process are (1) a set of sentences with three layers of annotation, which is used for extracting conversion rules, (2) the sentences with DS and PB annotation for which PS will be created, (3) rules manually-crafted for creating DS+. In order to create the desired PS, any information needed to form the PS has to be available in (1), (2), or (3).

6. Conclusion

An important question for treebank development is whether high-quality conversion from one representation to another representation (e.g., PS) is possible, assuming that annotation guidelines exist for both representations. In this study, we focus on DS-to-PS conversion, and demonstrate that conversion is possible only when certain conditions are met. We propose to use DS+ as a vehicle to bridge the gap between DS and PS analyses. When these conditions are met, PS can be created in two steps: creating DS+ from the input DS plus PB and generating PS from DS+. We then go over seven phenomena in the HUTB for which DS+ is needed, and show that creating DS+ is complex and cannot be fully automated. For future work, we will test our conversion process on the HUTB and evaluate the system performance on a small portion of the treebank where all three layers are manually annotated.

Acknowledgments This work is supported by NSF grants CNS-0751171 and CNS-0751213. We would like to thank the anonymous reviewers for helpful comments and our colleagues on the Hindi-Urdu Treebank Project for their support. Special thanks go to Annahita Farudi and Michael Tepper.

7. References

Rafiya Begum, Samar Husain, Arun Dhwaj, Dipti Misra Sharma, Lakshmi Bai, and Rajeev Sangal. 2008. Dependency annotation scheme for indian languages. In Proceedings of IJCNLP, Hyderabad, India.

Akshar Bharati, Vineet Chaitanya, and Rajeev Sangal. 1995. Natural Language Processing – A Paninian Perspective. Prentice-Hall of India.

Rajesh Bhatt, Owen Rambow, and Fei Xia. 2011. Linguistic phenomena, analyses, and representations: Understanding conversion between treebanks. In Proceedings of IJCNLP, pages 1234–1242, Chiang Mai, Thailand.

Aoife Cahill, Mairead McCarthy, Josef van Genabith, and Andy Way. 2002. Automatic Annotation of the Penn-Treebank with LFG F-Structure Information. In LREC 2002 Workshop on Linguistic Knowledge Acquisition and Representation - Bootstrapping Annotated Language Data.

Noam Chomsky. 1981. Lectures on Government and Binding. Dordrecht: Foris.

Michael Collins, Jan Hajič, Lance Ramshaw, and Christoph Tillmann. 1999. A statistical parser for czech. In Proceedings of ACL, pages 505–512.

Julia Hockenmaier and Mark Steedman. 2007. CCG-bank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. Computational Linguistics, 33(3):355–396.

Paul Kingsbury, Martha Palmer, and Mitch Marcus. 2002. Adding semantic annotation to the Penn Tree-Bank. In Proceedings of HLT, San Diego, CA.

Joakim Nivre. 2003. Theory-supporting treebanks. In In Proceedings of the TLT 2003 Workshop.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. Computational Linguistics, 31(1):71–106.

Martha Palmer, Rajesh Bhatt, Bhuvana Narasimhan, Owen Rambow, Dipti Misra Sharma, and Fei Xia. 2009. Hindi Syntax: Annotating Dependency, Lexical Predicate-Argument Structure, and Phrase Structure. In Proceedings of ICON, Hyderabad.

Fei Xia and Martha Palmer. 2001. Converting Dependency Structures to Phrase Structures. In Proceedings of HLT, San Diego, CA.

Fei Xia, Owen Rambow, Rajesh Bhatt, Martha Palmer, and Dipti Misra Sharma. 2009. Towards a multi-representational treebank. In The 7th International Workshop on Treebanks and Linguistic Theories (TLT-7), Groningen, Netherlands.

Arabic Treebank: from Phrase-Structure Trees to Dependency Trees

Maytham Alabbas, Allan Ramsay

School of Computer Science
University of Manchester
Manchester, M13 9PL, UK
{alabbasm, ramsay}@cs.man.ac.uk

Abstract

The aim here is to create a dependency treebank from a phrase-structure treebank for Arabic. Arabic has a number of characteristics, described below, which make it particularly challenging to any natural language processing (NLP) applications. We describe an encouraging semi-automatic technique for converting phrase-structure trees to dependency trees by using a head percolation table.

One of the most significant challenges here is the determination of the head of each subtree. We therefore examined different versions of the head percolation table to find the best priority list for each entry in the table. Given that there is no absolute measure of the 'correctness' of a conversion of a phrase structure tree to dependency form, we tested the various transformations by seeing how well a state-of-the-art dependency parser learnt the generalisations that were embodied by the converted trees.

1. Introduction

In recent years, dependency parsing has become widely used in machine translation, question answering, relation extraction and many other natural language processing (NLP) applications (Kübler et al., 2009) for a number of reasons.

- It provides uniform treatments for a wide range of typologically different languages, since it hides differences that arise from different surface word orders and emphasises the functional relations between words, which tend to be similar across languages.

- It can be useful for semantics. It supports compositional semantics, since it can be easier to attach compositional rules directly to lexical items than to assign them to large numbers of phrase-structure rules ; and it supports less formal approaches to semantics, e.g. via textual entailment (Dagan et al., 2005), since it is easier to generalise dependency trees as the basis of approximate inference rules than to use phrase structure rules for this purpose.

- It is possible to induce robust and fairly accurate dependency parsers, e.g. MSTParser (McDonald and Pereira, 2006; Nivre et al., 2010) and MALTParser (Nivre et al., 2006; Nivre et al., 2010), from treebanks of suitably annotated dependency trees.

Our ultimate goal is to develop a *textual entailment* (TE) system for Arabic (Alabbas, 2011). An efficient technique to check entailment between two sentences is using *Tree Edit Distance* (TED) (Bille, 2005) matching technique between dependency trees the two sentences (Kouylekov and Magnini, 2005; Heilman and Smith, 2010). We therefore need to be able to obtain dependency trees by parsing input Arabic texts, and hence we need dependency treebanks for training our dependency parsers (e.g. MSTParser or MALTParser).

The focus of the current paper is on the nature of the training data. Dependency grammar is a general framework, with a myriad major and minor variations. The distinctions between different versions of dependency grammar are usually debated in terms of their linguistic adequacy, but it is likely that different sets of dependency relations will produce different levels of accuracy when used for inducing parsers. In particular, we want to investigate the effects of different ways of turning the phrase structure trees in the Penn Arabic Treebank (PATB), which is the largest easily available set of training data for Arabic, into dependency trees.

The three best-known Arabic treebanking efforts are the Penn Arabic Treebank (PATB) (Maamouri and Bies, 2004), the Prague Arabic Dependency Treebank (PADT) (Smrž et al., 2008) and Columbia Arabic Treebank (CATiB) (Habash and Roth, 2009). PATB is a phrase-structure treebank, whereas the others are dependency treebanks. PATB and PADT are annotated with rich morphological information, very fine-grained part-of-speech (POS) tags, semantic roles, diacritisation and lemmas. This allows these treebanks to be used for training different natural language processing applications (e.g. tokenisation, POS tagging, diacritisation, morphological disambiguation and others) as well as parsing (Habash and Roth, 2009). In contrast, much of the detailed morphological and POS information is not provided in CATiB. The use of any of these treebanks as training data raises problems:

- The trees in PATB are phrase-structure trees, and hence are not directly useable for training dependency parsers.

- PADT uses very fine-grained POS labels, and there is no readily available tagger for assigning PADT labels (personal correspondence with the PADT designer).

- CATiB uses a very coarse-grained set of just 6 POS labels. In particular, all verbs are grouped under a single heading (i.e. `VRB`), with no distinction between present and past; and all nominals are also grouped together (i.e. `NOM`), with no distinction between nouns or numbers, or even between definite and indefinite forms. Other researchers have noted that dependency parsers can cope with very coarse-grained POS tags, but the loss of this information makes the trees in CATiB unsuitable for our underlying goal of inferring TE rules from trees.

Of the three, PATB seemed most suitable for our purposes, since we have taggers that can assign the tags used in PATB and the information it contains is what we need for inferring TE rules. We therefore need to convert PATB trees to dependency trees, which we do by using the standard phrase-structure to dependency transformation algorithm (given in Figure 1), using a percolation table to choose the head daughter in the phrase-structure tree.

This raises an interesting question: since PATB does not explicitly embody a notion of the 'head' of a tree, we have some freedom about how to construct the percolation table. Should the head of an NP be the main noun or the determiner? Should the head of a verbless sentence be the subject or the predicate? Should the head of a conjoined phrase be the conjunction itself or the head of the first conjunct? There are theoretical grounds for choosing one way of answering these questions rather than another, but they also have empirical consequences. It seems likely that one set of choices will provide the parsers with a more easily recognisable set of dependency relations than another. It may turn out that the parsers perform best with a set of trees that were obtained using a theoretically unappealing percolation table (e.g. we will see in Section 4. that choosing the first conjunct of a conjoined pair produces better results than choosing the conjunction itself, but it is much easier to obtain a formal interpretation from trees where the conjunction is the head (Gazdar, 1980)). It is, however, a straightforward matter to transform trees that based on one notion of conjunction to ones based on another, so from a practical point of view we want to find the percolation table that allows the parsers to produce the most accurate results.

2. The challenges of Arabic

The key problem for Arabic is that it is massively more ambiguous than English, for reasons described below.

Lexical ambiguity:

- Arabic is written without diacritics (short vowels), often leading to several ambiguities. This makes analysis of the language morphologically very complex and difficult (i.e. a single written form corresponding to as many as ten different lexemes). For instance, the following table shows the Arabic word علم *ςlm*[1] , which has 7 different reading with diacritics marks.

- Arabic also contains numerous clitic items (prepositions, pronouns and conjunctions), so that it is often difficult to determine just what items are present in the first place. For example the word والى *wAlý* can be analyzed as والي *wAly* "ruler", و+الى+ي *w+Alý+y* "and to me", و+ألي *w+Âly* "and I follow", و+آل+ي *w+Āl+y* "and my clan" or و+آلي *w+Āly* "and automatic" (Habash, 2010). Each of these cases has a different diacritisation.

Arabic diacritics word	Meaning
عِلْمٌ *ςilmũ*	knowledge
عَلَمٌ *ςalamũ*	flag
عَلِمَ *ςalima*	knew
عُلِمَ *ςulima*	is known
عَلَّمَ *ςal∼ama*	taught
عَلِّمْ *ςal∼im*	teach!
عُلِّمَ *ςul∼ima*	is taught

Table 1: ambiguity caused by the lack of diacritics.

Syntactic ambiguity:

- Arabic has a comparatively free word order, with VSO, VOS, SVO and OVS all being possible orders for the arguments of a transitive verb under appropriate conditions (Alabbas and Ramsay, 2011a).

- It is a pro-drop language. The subject can be omitted, leaving any syntactic parser with the challenge to decide whether or not there is an omitted pronoun in the subject position. For example, the Arabic sentence اكلت الدجاجة *Aklt AldjAjħ* "ate(feminine) the-chicken" has two different interpretations–"The chicken ate" or "(She) ate the chicken" (Attia, 2008).

- Nouns can be used as adjectives, or as possessive determiners (in so-called 'construct phrases'), with typically little inflectional morphology to mark such uses (Alabbas and Ramsay, 2011b). For instance,the Arabic construct phrase مفاتيح السيارة *mfAtyH AlsyArħ* has many comparables in English: "the keys of the car" or "the car's keys" or "the car keys" (Habash, 2010). In this example, the word السيارة *AlsyArħ* specifies, defines, limits or explains the particular identity of the word مفاتيح *mfAtyH*.

- The copula is omitted in simple positive *equational sentences*, so that a sequence of a noun and a predicative item (i.e. another noun, an adjective or a PP) may make a sentence. For instance, the Arabic equational sentence الولد في البيت (*Alwld fy Albyt*: "The boy (is) in the house.") has a PP predicate في البيت (*fy Albyt*: "in the-house").

3. From PATB to dependency trees

We used PATB Part 1 v3.0 as a resource that provides us with annotations of Arabic at different levels of structure (at the word, the phrase and the sentence levels). PATB is annotated for part-of-speech (POS), morphological disambiguation and for syntactic structure. It also provides diacritisations, empty categories, some semantic tags and

[1] The transcription of Arabic examples follows Habash-Soudi-Buckwalter (HSB) (Habash et al., 2007).

lemma choice. Because PATB is already tagged, the effects of lexical ambiguity are substantially reduced–we know which items are nouns and which are verbs, and we know about the clitics that have been attached to them. We still have no information about transitivity, which is itself a problem for parsing, but the gross lexical ambiguity that is brought about by the lack of diacritics in Modern Standard Arabic (MSA) has been eliminated. The PATB includes 734 stories representing 145,386 words (166,068 tokens after clitic segmentation; the number of Arabic tokens is 123,796). The sentences in PATB, which contains just over 5000 phrase-structure trees, are fairly long–the average sentence length is around 28 words per tree, with some trees containing 100+ words.

Converting phrase-structure trees to dependency trees is a straightforward task, so long as (i) you can identify the item in each subtree which contains the head, and (ii) there are no constructions with zero-heads. Assuming that these two conditions hold, the algorithm in Figure 1 will convert a phrase-structure tree to a dependency tree.

1. if you're looking at a leaf node, turn it into a tree with no daughters;
2. (a) otherwise, choose the subtree which contains the head: turn it into a dependency tree: call this D;
   (b) turn all the other subtrees into dependency trees, and add them as daughters of D.

Figure 1: From phrase-structure trees to dependency trees

The only difficult part of this algorithm, which has been discussed by (Xia and Palmer, 2001), is the selection of the subtree which contains the head. The approach we have taken to this task is to look for all the trees headed by a given label, and find all the labels for their subtrees. This gives us a list of labels for potential head daughters for each non-terminal label (a 'head percolation table' (Collins, 1997)). We then order these in terms of candidacy for being the head daughter, in terms of what we believe to be the correct dependency structure, and we use this preference order for *'choose the subtree which contains the head: turn it into a dependency tree'* in the above algorithm. For instance, in the head percolation table the entry (`S` left `VP`) means that the head child of an `S` is the first child of the `S` from the left with the label `VP`.

Here, the head percolation table is semi-automatically generated from PATB by grouping the related tags in one tag and then finding the possible heads for each one. After that, for each table's entry we order the possible heads manually according to its priority[2]. Then, the above algorithm is used to generate the dependency tree recursively. We used six dependency relations as an initial step to construct our treebank. These relations are: **SBJ** (subject), **OBJ** (object), **ROOT** (root), **COORD** (coordinate), **PX** (punctuation) and **DEP** (dependent). The full percolation table is available in a longer version of this paper which can be found at `http://www.cs.man.ac.uk/~alabbasm/`.

There are, however, a number of problems that arise when applying this algorithm to PATB:

- Very large numbers of Arabic sentences begin with the conjunction +و *w*+ "and". The most plausible reading is that this item implicitly conjoins the first clause in the current sentence to the previous sentence, and hence should be taken as its head. To take account of the difference between this use of conjunctions and their more normal use for joining two constituents we mark sentence-initial +و *w*+ "and" with a special tag (i.e. `ICONJ`) to prevent the parsers confusing it with more normal conjunctive uses of this item. By doing this, the parser's accuracy is improved by around 0.4% as shown in Section 4.

- PATB deals with free word-order by using traces, assuming in particular that the structure of SVO sentences is something like `[NP, [V, trace, O]]`, with the topicalised `NP` cancelling the `trace`. This doesn't really make sense within a dependency-based framework. The use of traces is antithetical to the basic idea of dependency grammar, namely that syntactic structure is determined by relations between *words*: a trace is not a word, and as such has no place in dependency grammar, at least as strictly conceived (e.g. by Hudson (1984)). We therefore systematically transform PATB so that traces are eliminated, with the topicalised `NP` treated as a proper constituent of the sentence.

- Arabic allows 'verbless' or 'equational' sentences, consisting of an `NP` and some kind of predication (another `NP`, a `PP`, an adjective). It is tempting to think of these constructions as containing a zero-copula (the fact that their negated forms do include an explicit copula supports this analysis). As noted before, we prefer to eliminate empty items, especially heads. We therefore have to choose whether to make the subject or the predicate of such sentences the head daughter. Our empirical findings show that making the subject the head gives better results than the reverse, as shown in Section 4.

- PATB uses a very fine-grained set of tags, which carry a great deal of syntactically relevant information (particularly case-marking). This tagset contains 306 tags, with for instance 47 tags for different kinds of verb and 44 for different kinds of noun. Unfortunately it is difficult to extract this information using a tagger. In particular, case-marking, and to a lesser extent number and gender marking, in Arabic is carried by diacritics which are unwritten in normal text. Thus the only way to extract this information is by making guesses based on the syntactic context. Given that the task of the parser is to determine the syntactic context, it is hard to see how reliable guesses about it can be made prior to parsing. Marton et al. (2010), for instance, note that adding the case-marking tags supplied by MADA (Habash et al., 2009b) actually decreases the accuracy, because the parser gives

[2] The consequences of using different versions of the percolation table are discussed in Section 4.

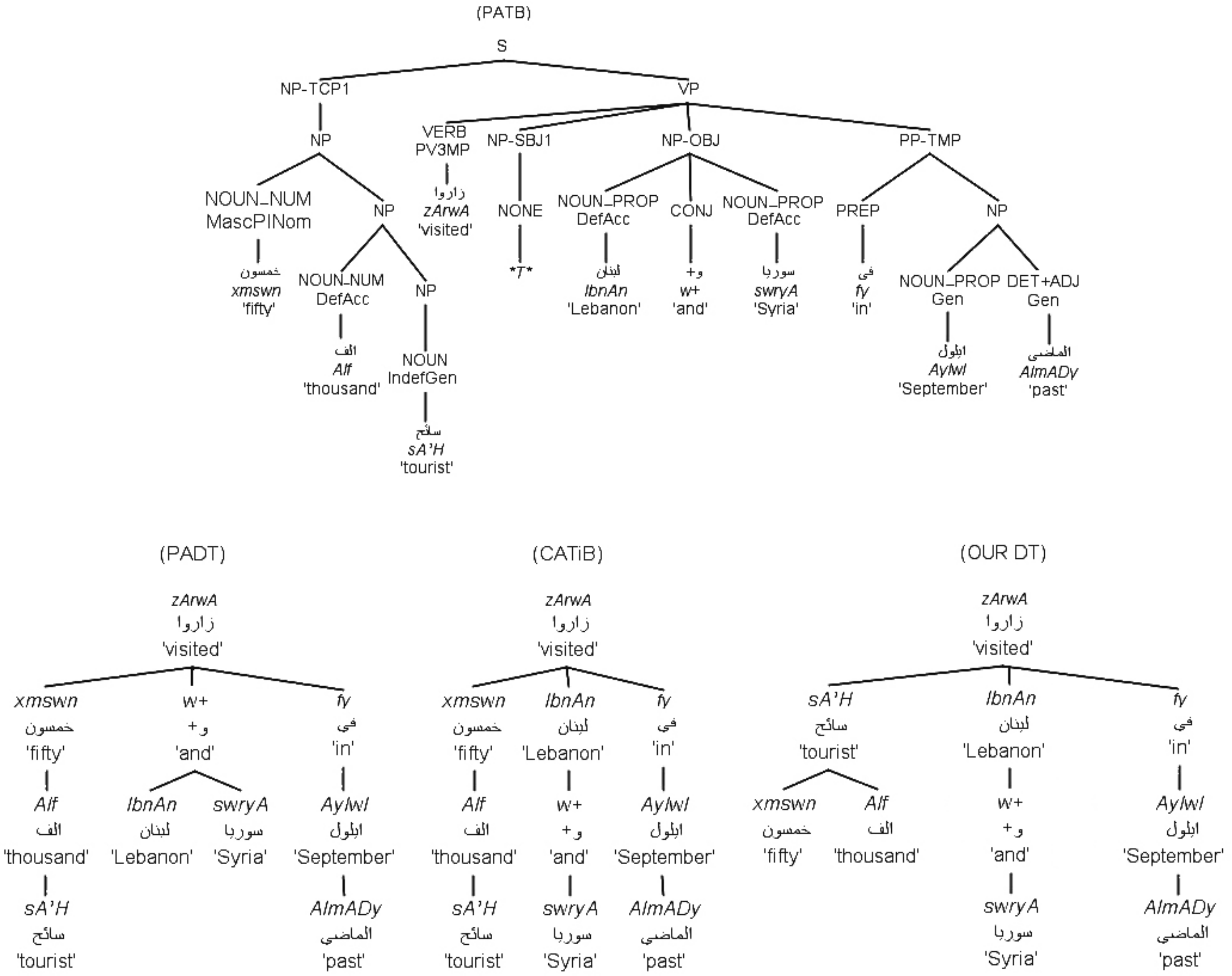


Figure 2: Comparing PATB phrase-structure and dependency format (without POS tags and labels) in PADT, CATiB and our preferred conversion for the sentence: خمسون الف سائح زاروا لبنان وسوريا في ايلول الماضي *xmswn Alf sA'H zArwA lbnAn w+swryA fy Aylwl AlmaDy* "Fifty thousand tourists visited Lebanon and Syria last September."(we deleted the POS tags and dependency relations from dependency trees to compare the trees structures only).

considerable weight to them, but the tagger only manages to assign them correctly in 86% of cases. A feature which is both significant and hard to determine is not something you want to depend on.

We therefore collapsed this set to a coarse-grained set with 39 distinct tags (e.g. the fine-grained tags `NOUN+CASE_DEF_ACC`, `NOUN+CASE_DEF_GEN`, `NOUN+CASE_DEF_GEN+POSS_PRON_3MS`, `NOUN+ CASE_DEF_NOM` grouped to `NOUN`). We can tag this set to just over 96% accuracy, which is comparable to the performance of other taggers (Al Shamsi and Guessoum, 2006; AlGahtani et al., 2009; Hadj et al., 2009; Diab et al., 2004) on this kind of tagset, whereas we only achieve 91% with the full 306 tags (similarly MADA obtains 96.6% on the coarse-grained tagset but only 93.6% on the fine-grained (356-element) one).

The dependency trees to be used by the parsers also have to be labelled with functional relations. Since dependency grammar is entirely concerned with relations between words, any information beyond simple constituency has to be encoded in the labels on the relations. Such relations tend not be explicitly marked in phrase-structure trees, since they are often implied by the label of the local tree (to put it simply, it is not necessary to mark the `NP` daughter of a sentence as its subject, because this is implicit in the rule that says that a sentence may be made out of an `NP` and a `VP`). We therefore have to *impose* a set of functional relations. We cannot use a very fine-grained set of labels here, because the information in PATB simply does not provide enough information. The only labels we can assign with any degree of confidence relate to conjunctions, and to the subject and object of the verb.

There is no consensus about the set of relations to be used in dependency grammar. Some authors, for instance take it that the auxiliary dominates the verb it is associated with, while others take the opposite view. Similarly, coordinating conjunctions are sometimes treated as heads and sometimes as modifiers. We investigated three issues:

Is the determiner or the noun the head of an NP? Where an `NP` contains a determiner, some theories (e.g. categorial grammar) treat the determiner as the head and others (e.g. HPSG) treat it as a dependent of the noun. Arabic makes less use of determiners than some other languages, since there is no indefinite article and the definite article is treated as a part of the noun to which it is attached, but it does have numbers and demonstrative determiners, and choosing whether these are heads or dependents may make a difference.

Is the subject or the predication the head of verbless sentences? Arabic verbless sentences are widely regarded as containing an invisible 'zero' copula. If the copula were present, it would be the head, but since it is missing then either the subject or the predication will have to be chosen. Which of these makes the parser perform better?

What is the head of a conjoined phrase? The choice of how to deal with conjunctions in dependency grammar is widely disputed, with reasonable arguments being put forward on both sides. The decision to treat the number as a modifier on the noun in the subject NP coincides with the treatment of numbers as determiners in grammatical frameworks such as GPSG (Gazdar, 1985), HPSG (Pollard and Sag, 1994) and so on, and indeed with the treatment in the original phrase-structure tree in PATB.

We will use the sentence in (1), from (Habash et al., 2009a), to highlight the differences between our conversion and the other Arabic dependency treebanks.

(1) خمسون الف سائح زاروا لبنان وسوريا في ايلول الماضي

xmswn Alf sA'H zArwA lbnAn w+swryA fy Aylwl AlmaDy

"Fifty thousand tourists visited Lebanon and Syria last September."

As can be seen in Figure 2, our dependency tree is different from the others in the sentential part خمسون الف سائح *xmswn Alf sA'H* "Fifty thousand tourists", because we considered the NOUN as the head not NOUN_NUM and the others are dependent, whereas both PADT and CATiB considered the first NOUN_NUM as the head. Furthermore, we follow the CATiB of the conjoined NP لبنان وسوريا *lbnAn w+swryA* "Lebanon and Syria", where the head is the first noun, with the conjunction as its sole daughter and the second noun as a daughter of the conjunction.

It should be noted that this treatment of conjunction cannot be obtained from the original PATB tree by the algorithm given above. The conjoined phrase لبنان وسوريا (*lbnAn w swryA*: "Lebanon and Syria") has three constituents– the conjunction and two NPs. Specifying that the conjunction has priority over any other constituent in the percolation table would produce the subtree given for this phrase in PADT. Specifying that the conjunction has lower priority than one of the NPs would produce a subtree with either لبنان (*lbnAn*: "Lebanon") (or possibly سوريا (*swryA*: "Syria")) as its head and the conjunction and the other noun as daughters. There is no way to get the conjunction as a daughter of لبنان (*lbnAn*: "Lebanon") and then سوريا (*swryA*: "Syria") as a daughter of the conjunction by using the algorithm in Figure 1. In order to obtain trees of this kind, we transform the original phrase structure tree so that structures of the form T1 become T2 in Figure 3. If we then assign DUMMY the lowest possible priority in the percolation table and CONJ the highest possible priority for the entry of DUMMY in the table we obtain CATiB-style trees for coordinated structures.

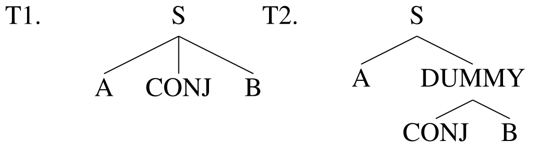


Figure 3: Reconstruct coordinated structures

We extend this treatment to cover cases where we have a complex coordinated expression where several of the conjuncts are linked by commas in addition to explicit conjunctions, converting phrases structure trees like the initial tree T1 in Figure 4 to the tree D1 in this figure instead of D2. This is the treatment used in experiment (7) below.

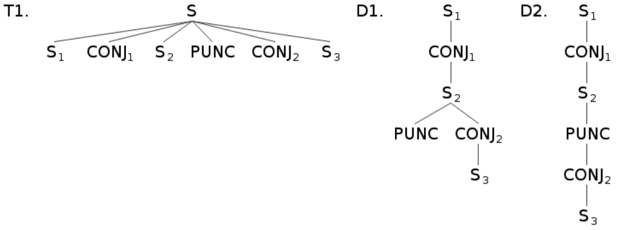


Figure 4: Complex coordinated structures

4. Experimental Results

To check the effectiveness of our conversion version of PATB to dependency tree format, we used it to train a state-of-the-art dependency parser, namely MSTParser. The aim here was to see how changes in the structure of the trees affect the performance of the parser: different ways of organising trees may be better or worse at capturing the regularities encapsulated in the original phrase-structure trees, or they make the cues that the parser depends on easier or harder to see.

Large and complex Arabic sentences are used in training and testing the parser (some sentences in each set contain 100 or more words). MSTParser is trained on 4000 sentences and tested on 1000 sentences. We examined 8 different ways of converting phrase-structure trees to dependency trees, in addition to two baselines. These variations were obtained by changing the priority of items in each entry in the percolation table, and by applying systematic transformations to PATB trees before converting them to dependency trees. The two baselines were obtained by selecting head daughters at random, and by reversing the percolation table that produced the best result when used normally.

The main experiments were as follows:

1. CONJ always has higher priority than anything else (so coordinating conjunctions are the head of the coordinated phrase), as in PADT; nouns have higher priority inside NPs than determiners (opposite of PADT and CATiB); the head of a verbless sentence is the subject (opposite of CATiB).

Version#	Items order for each entry in the head percolation table	LA	UL
	Random head (baseline 1)	20.2%	20.6%
	Inverse of best table (baseline 2)	73.5%	74.3%
1	CONJ has higher priority than other items	80.1%	81.5%
2	Split CONJ into CONJ and ICONJ	80.5%	81.9%
3	ICONJ,CONJ have lowest possible priority	80.8%	82.2%
4	Same as (3) but determiners above nouns in NPs	80.3%	81.7%
5	Same as (2) but CATiB-style conjunctions	82.5%	83.9%
6	Same as (5) but determiners above nouns in NPs	82.3%	83.7%
7	**Same as (5) but commas in conjunctions not treated as conjunctions (i.e. tree D1 in Figure 4, not tree D2)**	**82.8%**	**84.2%**
8	Same as (7) but head of verbless sentence is predication	82.3%	83.8%

Table 2: LA and UL accuracies for parsing, different head percolation table entry orders and treatments of coordination.

2. Same as (1), but sentence initial conjunctions are given a separate tag as ICONJ.

3. ICONJ,CONJ have lower priority than anything else: everything else as in (2)

4. Same as (2) but determiners have higher priority than nouns in NPs.

5. CATiB-style treatment of conjunctions (as discussed above). Everything else as in (2).

6. Same as (5) but determiners have higher priority than nouns in NPs.

7. Same as (5), but commas in coordinated expressions are not treated as though they were conjunctions (i.e. tree D1 in Figure 4, not tree D2).

8. Same as (7), but the head of a verbless sentences is assumed to be its predication rather than its subject.

The *labelled attachment score* (LA), i.e. the percentage of tokens with correct head and dependency relation, and *unlabelled accuracy* (UA), i.e. the percentage of tokens with correct head, for the various transformations of PATB to dependency tree format are shown in Table 2.

Varying the treatments of conjunction, NPs and verbless sentences can affect the performance of the parser by around 2.7%. The individual changes are not dramatic, but then each such change only affects one set of relations: there are, for instance, only around 800 determiners in the test set, out of a total of just over 25000 words. So if every instance of a determiner was labelled correctly in (3) and incorrectly in (4) there would only be a 3% difference in the total score, so the actual swing of 0.5% between these two cases is quite significant.

5. Conclusions

The results above show that changing the rules for converting phrase structure trees to dependency trees can affect parser accuracy even with the same parser and the same training and testing data. We obtained LA of 82.8% and UA of 84.2% for the head percolation table version (7) with CATiB-style treatment of conjunction, compared with LA ranging from 80.1% to 82.5% and UA from 81.5% to 83.9% for other percolation tables and treatments of conjunction. It may be that one form of tree captures the underlying regularities better than another, or it may be that one form makes the contextual clues more visible to the parser than another. Consider, for instance, the coordinated expression in (1): the association between the verb زاروا (*zArwA*: "visit") and the name لبنان (*lbnAn*: "Lebanon") is likely to be stronger than that between the verb and و (*w*: "and"), so the link between the verb and the name is more likely to be learnt and retrieved when the dependency trees are structured as in CATiB than the link between the verb and the conjunction would be if the trees followed the PADT approach to conjunction.

Where two dependency formats are intertranslatable (e.g. taking either the subject or the predicate to be the head of an equational sentence), then it makes sense to use the version which produces the optimal parser output, since under these circumstances it can be translated to the alternative if that seems preferable for some task. In particular, transforming coordinated expressions so that the first conjunct becomes the head is reversible, even in cases like *young men and women* which have multiple interpretations *(young men) and women*, *young (men and women)*. Converting the first of these so that *'men'* becomes the head leaves *young* and *and* as daughters, whereas the second will have *and* as the sole daughter of *man*, and *young* and *women* as daughters of *and*. If the new tree is appropriately labelled then there is no problem with re-transforming it back to the original.

Appendix A shows UA and LA for each class of word, where column 3 ('right link': UA) shows how many times words of that class have been assigned as daughters in links with the right head, column 5 ('right label') shows how many times words of the class have been assigned as daughters in links with the right label, and column 7 ('both right': LA) shows how many times they have been assigned the right head and the right label[3]. The most striking thing about this table is the score for prepositions: these are common (12% of the total number of words) and the parser does very poorly at finding their heads (73% UA). This is

[3] This table contains just 29 entries–several of the full set of 39 tags do not appear as heads in the test set, either because they never appears as heads at all or because they are rare and hence happen not to occur in the test set (e.g. PUNC and LATIN).

unsurprising, since getting the right head for a preposition is equivalent to solving the PP-attachment problem, which is notoriously difficult. In general, we need very large amounts of training data before straightforward statistical techniques can detect the lexical patterns that underly successful strategies for PP-attachment. When we start writing TE rules using the trees that the parser obtains from our text:hypothesis pairs, we will use the confidence measures for particular kinds of link to as part of our confidence measure for the inferred rules.

6. References

F Al Shamsi and A Guessoum. 2006. A hidden Markov model-based POS tagger for Arabic. In *Des Journées internationales d'Analyse statistique des Données Textuelles (JADT-8)*, pages 31–42, Besançon.

M. Alabbas and A. Ramsay. 2011a. Evaluation of Combining Data-Driven Dependency Parsers for Arabic. In *Proceeding of 5th Language & Technology Conference: Human Language Technologies(LTC'11)*, pages 546–550.

M. Alabbas and A. Ramsay. 2011b. Evaluation of Dependency Parsers for Long Arabic Sentences. In *Proceeding of International Conference on Semantic Technology and Information Retrieval (STAIR'11)*, pages 243–248. IEEE.

M. Alabbas. 2011. Arbte: Arabic Textual Entailment. In *Proceedings of the Second Student Research Workshop associated with RANLP 2011*, pages 48–53, Hissar, Bulgaria. RANLP 2011 Organising Committee.

S AlGahtani, W J Black, and J McNaught. 2009. Arabic part-of-speech tagging using transformation-based learning. In *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, pages 66–70, Cairo.

Mohammed Attia. 2008. *Handling Arabic Morphological and Syntactic Ambiguity within the LFG Framework with a View to Machine Translation.* Ph.D. thesis, Manchester University.

P. Bille. 2005. A survey on tree edit distance and related problems. *Theoretical computer science*, 337(1-3):217–239.

M. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 16–23. Association for Computational Linguistics.

I Dagan, B Magnini, and O Glickman. 2005. The PASCAL recognising textual entailment challenge. In *Proceedings of Pascal Challenge Workshop on Recognizing Textual Entailment*.

M Diab, K Hacioglu, and D Jurafsky. 2004. Automatic tagging of Arabic text: from raw text to base phrase chunks. In *Proceedings of NAACL-HLT 2004*, pages 149–152, Boston.

G Gazdar. 1980. A cross-categorial semantics for coordination. *Linguistics & Philosophy*, 3:407–409.

G. Gazdar. 1985. *Generalized phrase structure grammar*. Harvard University Press.

N. Habash and R.M. Roth. 2009. Catib: The columbia arabic treebank. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 221–224. Association for Computational Linguistics.

N. Habash, A. Soudi, and T. Buckwalter. 2007. On arabic transliteration. *Arabic Computational Morphology*, pages 15–22.

N. Habash, R. Faraj, and R. Roth. 2009a. Syntactic annotation in the columbia arabic treebank. In *Proceedings of the 2nd International Conference on Arabic Language Resources and Tools*, pages 125–132, Cairo. The MEDAR Consortium.

N. Habash, O. Rambow, and Roth R. 2009b. MADA+TOKAN: A Toolkit for Arabic Tokenization, Diacritization, Morphological Disambiguation, POS Tagging, Stemming and Lemmatization. In *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, Cairo. The MEDAR Consortium.

N. Habash. 2010. *Introduction to Arabic Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Y El Hadj, I Al-Sughayeir, and A Al-Ansari. 2009. Arabic part-of-speech tagging using the sentence structure. In *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, pages 241–245, Cairo. The MEDAR Consortium.

M. Heilman and N. A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1011–1019. Association for Computational Linguistics.

R Hudson. 1984. *Word Grammar*. Basil Blackwell, Oxford.

M. Kouylekov and B. Magnini. 2005. Recognizing textual entailment with tree edit distance algorithms. In *Proceedings of the First Challenge Workshop Recognising Textual Entailment*, pages 17–20.

S. Kübler, R. McDonald, and J. Nivre. 2009. *Dependency parsing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

M. Maamouri and A. Bies. 2004. Developing an Arabic treebank: Methods, guidelines, procedures, and tools. In *Proceedings of COLING*, pages 2–9.

Y. Marton, N. Habash, and O. Rambow. 2010. Improving arabic dependency parsing with lexical and inflectional morphological features. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 13–21. Association for Computational Linguistics.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, volume 6, pages 81–88.

J. Nivre, J. Hall, and J. Nilsson. 2006. MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, volume 6, pages 2216–2219.

J Nivre, L Rimell, R McDonald, and C Gómez-Rodríguez.

2010. Evaluation of dependency parsers on unbounded dependencies. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 833–841, Beijing.

C.J. Pollard and I.A. Sag. 1994. *Head-driven phrase structure grammar*. Chicago University Press.

O. Smrž, V. Bielicky, I. Kouřilová, J. Kráčmar, J. Hajič, and P. Zemánek. 2008. Prague arabic dependency treebank: A word on the million words. In *Proceedings of the Workshop on Arabic and Local Languages (LREC 2008)*, pages 16–23, Morocco.

F. Xia and M. Palmer. 2001. Converting dependency structures to phrase structures. In *Proceedings of the first international conference on Human language technology research*, pages 1–5. Association for Computational Linguistics.

Appendix A: UA and LA by POS tag

Accuracy	words	right link (UA)	%	right label	%	right link & label (LA)	%
total	25643	21590	84%	24013	94%	21231	83%
NOUN	4459	3719	83%	3898	87%	3591	81%
PREP	4075	2981	73%	4005	98%	2979	73%
DET+NOUN	3567	3210	90%	3209	90%	3080	86%
NOUN_PROP	2462	2142	87%	2282	93%	2107	86%
DET+ADJ	2023	1819	90%	1929	95%	1818	90%
PV	1447	1302	90%	1424	98%	1302	90%
CONJ	967	704	73%	966	100%	704	73%
ADJ	927	757	82%	846	91%	753	81%
NUM	786	653	83%	755	96%	649	83%
ICONJ	750	721	96%	745	99%	721	96%
IV	746	596	80%	712	95%	583	78%
SUB_CONJ	700	617	88%	698	100%	617	88%
DET+NOUN_PROP	514	444	86%	457	89%	434	84%
POSS_PRON	495	446	90%	457	92%	436	88%
PRON	373	328	88%	340	91%	319	86%
REL_PRON	318	307	97%	315	99%	307	97%
PART	299	241	81%	278	93%	241	81%
DEM_PRON	194	183	94%	187	96%	181	93%
ADV	128	90	70%	122	95%	89	70%
NO_FUNC	109	66	61%	99	91%	64	59%
FUT+IV	78	63	81%	74	95%	63	81%
PVSUFF_DO	76	76	100%	72	95%	72	95%
ABBREV	56	41	73%	56	100%	41	73%
IVSUFF_DO	46	45	98%	41	89%	41	89%
REL_ADV	27	23	85%	26	96%	23	85%
DET	14	11	79%	14	100%	11	79%
INTERJ	5	3	60%	4	80%	3	60%
CV	1	1	100%	1	100%	1	100%
SUB	1	1	100%	1	100%	1	100%

What We Have Learned from Sofie: Extending Lexical and Grammatical Coverage in an LFG Parsebank

Gyri Smørdal Losnegaard*, **Gunn Inger Lyse***, **Martha Thunes***, **Victoria Rosén***§, **Koenraad De Smedt***, **Helge Dyvik***§, **Paul Meurer**§

University of Bergen* and Uni Research§
Bergen, Norway
gyri.losnegaard@lle.uib.no, gunn.lyse@lle.uib.no, martha.thunes@lle.uib.no, victoria@uib.no,
desmedt@uib.no, paul.meurer@uni.no, dyvik@uib.no

Abstract

Constructing a treebank as a dynamically parsed corpus is an iterative process which may effectively lead to improvements of the grammar and lexicon. We show this from our experiences with semiautomatic disambiguation of a Norwegian LFG parsebank. The main types of grammar and lexicon changes necessary for achieving improved coverage are analyzed and discussed. We show that an important contributing factor to missing coverage is missing multiword expressions in the lexicon.

1. Introduction

The INESS project[1] (2010–2015) is building a highly detailed treebank for Norwegian by parsing corpora with the NorGram LFG grammar (Dyvik, 2000; Butt et al., 2002). A major challenge for the automatic analysis of corpora is incomplete coverage by the grammar and lexicon, in particular related to the phenomenon of multiword expressions. These are poorly documented in most languages, including Norwegian. Although NorGram has an extensive lexicon, few multiword expressions are included. In our work with semiautomatic disambiguation of corpora we have noted that multiword expressions missing from the lexicon cause many gaps in coverage, a problem that often may be solved by simply adding the relevant expression to the lexicon.

In this paper we present a study of a small subcorpus in the INESS Norwegian treebank. This study diagnoses some of the reasons for missing coverage, and we show that a large proportion of the problems are caused by missing multiword expressions.

2. The parsebanking method in INESS

The parsebanking method used in the INESS project involves parsing, disambiguation, and grammar and lexicon development in an iterative cycle. This method was perhaps first developed for HPSG grammars in the LinGO project (Oepen et al., 2004). Overviews which describe this type of approach are found in Branco (2009) and Bender et al. (2011).

In our approach, a corpus is first parsed automatically using the Xerox Linguistic Environment (XLE) (Maxwell and Kaplan, 1993) and the NorGram LFG grammar. LFG analyses provide two separate but parallel levels of syntactic analysis. There is a constituent structure (c-structure) in the form of a context-free phrase structure tree, and a functional structure (f-structure), an attribute–value matrix with information on grammatical features and syntactic functions.

Since automatic parsing with a handwritten grammar produces many analyses, efficient disambiguation is necessary. This is done using discriminants, as described in more detail elsewhere (Carter, 1997; Oepen et al., 2004; Rosén et al., 2007; Rosén et al., 2009). After disambiguation has been achieved, it becomes apparent whether the intended analysis is present. When this is not the case, the annotators try to diagnose the reason for the problem. This may be done by experimenting with XLE-Web (Rosén et al., 2005), for instance by changing or deleting words or phrases to check if modified sentences get an analysis, and suggesting changes to the grammar or lexicon. After necessary changes have been made in the grammar and lexicon, the corpus is reparsed.

After each reparsing, the corpus is automatically disambiguated by means of the reapplication of cached discriminants. In some cases the previously used discriminant choices are no longer sufficient to fully disambiguate the sentence due to the changes made in the grammar and lexicon. In such cases, some additional discriminant choices must be made by the annotators.

A major advantage of this approach is that the analyses in the treebank are always in accordance with the grammar, and coverage may be improved in a principled way.

3. Study of the Norwegian Sofie treebank

For this study we have investigated the first 255 sentences of the novel *Sofies verden* [Sophie's World] (Gaarder, 1991) to find out what the major problem types are that need to be addressed in order to achieve coverage of a subcorpus. The 255 sentences were initially parsed without prior examination of their vocabulary. Then followed several rounds of disambiguation, problem diagnosis, grammar and lexicon changes, and reparsing. The results of parsing in the first and fourth rounds are shown in Table 1 (all numbers are percentages).

version	gold	not gold	frag	0 sol	no parse
1	26	21	26	5	22
4	78	2	4	14	2

Table 1: Initial and subsequent parse results

[1] http://iness.uib.no

In the first round, 73% of the sentences received analyses (the categories 'gold', 'not gold' and 'frag(ment)', while 27% received no analyses (the categories '0 sol(utions)' and 'no parse').

There were full analyses for 47% of the sentences, while 26% of the sentences had the intended analysis (gold) without any intervention in the grammar or lexicon.

Fragment analyses occur when the grammar is unable to assign a global analysis to the sentence, and instead returns the analyses of the maximal phrases which it has been able to parse. Fragment analyses may indicate shortcomings of the grammar, or they may indicate strings deviating from the grammatical norms of the language (Rosén and De Smedt, 2007). In the latter case the fragment analyses are considered as valuable information worth storing. An example from the corpus is the sentence - *Det ... det er en hemmelighet* 'It ... it is a secret', with the dots signaling hesitation on the part of the speaker. The fragment analysis is shown in Figure 1, indicating the analyzable chunks of the string.

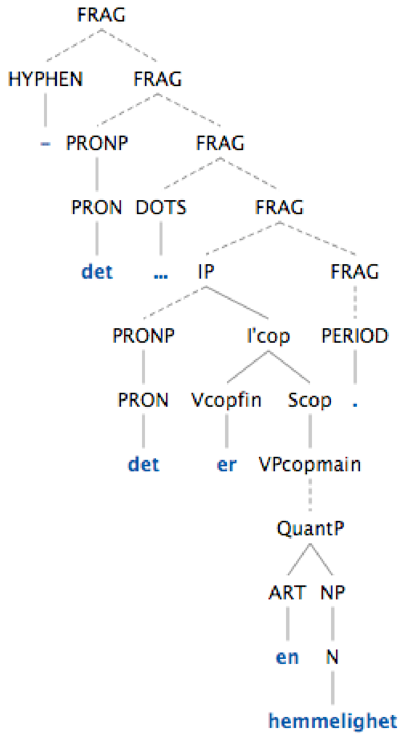


Figure 1: Fragment analysis

The category '0 solutions' means that the parser terminated, but found no analysis in accordance with the grammar, while the category 'no parse' means that the parser didn't terminate within the time and space parameters set for it. As we will show, the first category was significantly reduced as a result of the interaction between annotators and grammar developer, while the second category grew as a result of increased grammar complexity, leading to occasional explosions in local ambiguities. The latter problem will be addressed later in the project by devising methods for shallow preprocessing of sentences, thus reducing local ambiguity.

4. Problem analysis

We have done an in-depth study of the parsed sentences that were missing the intended analysis in version 1 but that did receive the intended analysis in version 4. In particular, we have studied the interventions that were necessary in order to produce the desired analysis. We have concentrated on the sentences that originally had full analyses rather than fragments, since these have received the most attention so far in disambiguation and problem diagnosis. For a small number of these sentences (2%) it was not possible to ascertain why they did not get the intended analysis originally. This could be because the problem diagnosis was not recorded. For most of the sentences, we have, however, identified the problem or problems; some sentences had multiple problems. For this set of sentences (the 21% 'not gold' in version 1) the problems may be analyzed into two main categories: grammar problems (29%) and lexicon problems (71%). The two most prevalent types of lexicon problems are multiword expressions and lexical category problems, which make up 41% and 31% of the lexicon problems respectively. In the following sections we discuss the various problem types.

4.1. Grammar problems

Under the category of grammar problems we have considered various instances of shortcomings in the rule component of the grammar. In these cases it is necessary to extend the grammar by including types of constructions that have not yet been covered, and in order to solve such problems, the grammar writer must face the challenge of describing exactly the necessary classes of constructions, while avoiding the introduction of changes that may cause the grammar to overgenerate. Required changes may involve the writing of new phrase structure rules, as well as the modification of existing rules. Reported grammar problems may be illustrated by two examples.

(1) *Et menneske måtte da være noe mer*
a human must then be something more
enn en maskin?
than a machine
'A human then had to be something more than a machine?'

In (1) the original analysis of the comparative construction *noe mer enn en maskin* was not satisfactory. The quantifier *noe* was recognized as an adverb of degree (ADVdeg), giving the meaning 'somewhat more', and the expression *noe mer* was parsed as a quantifier phrase (QP), as shown in the c-structure in Figure 2. The intended analysis was achieved by modifying the rule describing quantifier phrases, and the modification involved allowing for recursivity in phrases of this category. In simplified terms, recursivity was introduced on the condition that the left-most Q must be a form of the quantifier *noe*. This resulted in a new analysis of the phrase *noe mer enn en maskin* where *noe* is correctly parsed as a quantifier, and the entire expression is assigned the hierarchical structure shown in Figure 3.

(2) *Uansett hva Sofie gjorde, gjorde hun*
regardless what Sofie did did she
akkurat det samme.
exactly the same
'No matter what Sofie did, she did just the same.'

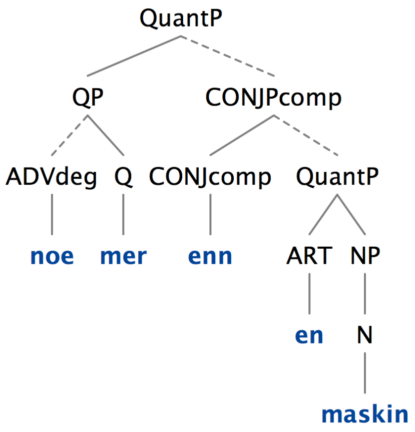


Figure 2: Incorrect analysis of *noe mer*

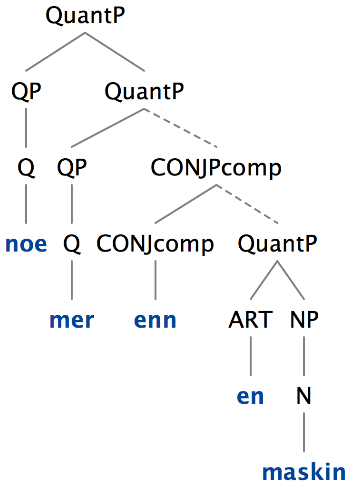


Figure 3: Correct analysis of *noe mer*

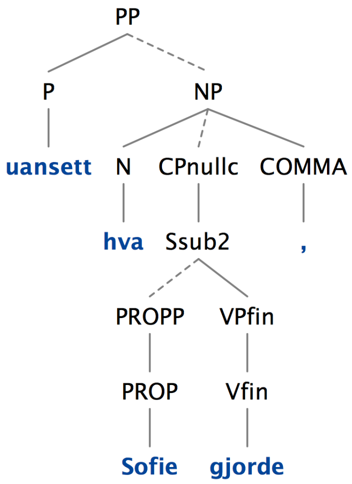


Figure 4: Incorrect analysis of *hva Sofie gjorde*

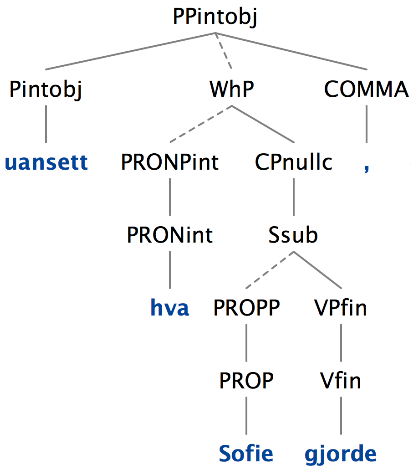


Figure 5: Correct analysis of *hva Sofie gjorde*

In (2) an unsatisfactory analysis of the expression *uansett hva Sofie gjorde* was amended by introducing a new phrase structure rule allowing certain prepositions to take interrogative constructions as their object. Originally, the expression *hva Sofie gjorde* was wrongly recognized as a noun phrase with a clausal postmodifier, and in that analysis the phrase was treated as the argument of the preposition *uansett*, as shown in Figure 4. Here the grammar update involved creating a special rule for prepositional phrases taking interrogative phrases as object (PPintobj), as shown in Figure 5, and the application of this rule is restricted to a limited set of prepositions, of which *uansett* is an example. Solving this problem also involved adding to the lexicon a new reading of *uansett* with the lexical category Pintobj. In the new analysis we achieved a satisfactory parse of the interrogative expression *hva Sofie gjorde*, where *hva* is correctly recognized as an interrogative pronoun.

In NorGram the analysis of punctuation is incorporated in the rule component of the grammar. Hence, cases where sentences have not been parsed because the parser has not recognized specific uses of punctuation marks may in the context of the present study be regarded as special cases of grammar problems. Among the challenges encountered in automatic parsing of authentic text is the correct analysis of various kinds of punctuation marks. Since the orthographic conventions governing the use of punctuation marks are not always very clear or generally agreed upon, it is not a trivial task to handle the various possible uses of different marks. In the Sofie treebank we have observed several cases where sentences have not been parsed successfully because particular ways of using specific punctuation marks were not covered by the grammar rules. Some examples of the use of dashes may illustrate this type of challenge.

(3)	*Der*	*lå*	*et*	*prospektkort*	*også*	—	*med*
	there	lay	a	postcard	also		with
	bilde	*av*	*en*	*sydlig*	*strand*		
	picture	of	a	southerly	beach		

‘There was a postcard too — with a photo of a southern beach.’

(4)	*Joda*	—	*det*	*var*	*ekte*	*nok,*	*med*	*både*
	yes		it	was	genuine	enough	with	both
	frimerke	*og*	*stempel.*					
	stamp	and	postmark					

‘Oh yes — it was real enough, with both a stamp and a postmark.’

(5) *Sofie stirret ned i asfalten — og opp*
Sofie stared down in asphalt:the and up
på venninnen.
on girlfriend:the
'Sofie stared down at the asphalt — and back at her friend.'

In these three sentences the effect of the dash is to create a short pause, and in the cases of (4) and (5) this pause puts a slight emphasis on the expressions following the dashes. In the original treebank version the parser produced a fragment analysis for each of these sentences. With respect to (3), this was the case because the dash prevented the parser from recognizing the prepositional phrase *med bilde av en sydlig strand* as being part of the sentence, and in the case of (5), the initial word *joda* was not incorporated in the sentence because of the following dash. As regards sentence (5), it contains two coordinated PPs, *ned i asfalten* and *opp på venninnen*, which are linked together by the conjunction *og*, but the parser fails to recognize the coordination because of the dash preceding the conjunction. In each of these cases, substituting a comma for the dash gave a full analysis of the sentence, and the problems were solved by amending the rule component of the grammar to allow dashes on a par with commas in these types of syntactic positions.

These three examples illustrate a few points that are common to several cases of punctuation problems. Firstly, to solve such problems it may be necessary to modify several grammar rules; in relation to the dash, rules applying to sentence as well as to verb phrase level were involved. Secondly, because the use of several types of punctuation marks, such as dashes, colons, and quotes, may be rather idiosyncratic, i.e. governed by individual authors' preferences, fairly ad hoc solutions may be required by the grammar developer in order to account for the various uses of different punctuation marks. The practical consequence of the latter point is that the punctuation found in a given corpus may to a great extent determine the ways in which the grammar writer chooses to solve the punctuation problems at hand.

4.2. Problems with multiword expressions

The largest group of lexicon problems encountered in our analysis had to do with multiword expressions (MWEs), which can roughly be defined as "idiosyncratic interpretations that cross word boundaries (or spaces)" (Sag et al., 2002). MWEs challenge the division between grammar and lexicon in linguistic theory due to the fact that they are lexicalized, but they may show variation at the morphological, syntactic and semantic levels.

Varying in terms of syntactic flexibility, MWEs can be grouped into the subcategories *fixed*, *semi-fixed* and *syntactically-flexible* expressions (Baldwin and Su Nam Kim, 2010; Sag et al., 2002). Semi-fixed and syntactically flexible expressions pose the biggest challenges in automatic analysis because they inflect, take internal modification or in other ways realize morphosyntactic variation. Different morphosyntactic categories of MWEs tend to adhere to different flexibility categories; for instance, verbal constructions are generally syntactically flexible. Depending on their flexibility, MWEs can be treated as words with spaces (fixed expressions) or as constructions. Constructions may either deviate from the morphosyntactic regularities of the language or be fully compositional in the sense that the rules of the grammar capture their syntactic and semantic properties satisfactorily. In these cases we have left them 'as they are' and not explicitly marked them as MWEs. Sag et al. (2002) single out four main problems related to the representation of MWEs in NLP systems. Treating MWEs as a problem of the lexicon poses a *flexibility problem* because we fail to capture morphosyntactic flexibility such as internal modification, or cases where some constituents inflect while others do not. Listing every single MWE in the lexicon also leads to a *lexical proliferation problem*: we lose out on generalities such as families of verbal constructions. By treating MWEs as a problem of grammar, the application of general compositional methods will lead to an *overgeneration problem* because the grammar will allow for anticollocations and other unacceptable constructions. Finally, *idiomaticity problems* may occur because grammar rules, working on sentence level, cannot distinguish between literal and figurative meanings.

In their diagnosis, the annotators reported all instances of possible MWEs that might have caused problems. The decision on which expressions should be implemented as MWEs and which should get a compositional analysis was made for each instance. With lexicon overpopulation in mind, we have as far as possible tried to analyze light verb constructions compositionally, although their deviating semantics clearly indicate what Baldwin and Kim refer to as MWEhood (Baldwin and Su Nam Kim, 2010). We cannot be certain that similar expressions have always been analyzed the same way, but as we gain experience in this early phase of annotation and get a better overview of the different types of MWEs, we will eventually be able to have a more principled approach to their identification and implementation.

For this study we have chosen to classify MWEs as a lexicon problem because of the practical implications of their analysis within the LFG framework: all implementations of MWEs have so far taken place in the lexicon, and not at rule level. Their syntactic variation is still fully accounted for by the grammar.

We have distinguished between two types of MWEs: verb frames and other MWEs. MWE verb frames include light verb constructions (*ta slutt* 'end'), particle verbs (*se ut* 'look'), and selected prepositions (*vite om* 'know about', 'be aware of'). Light verb constructions are verbs with noun complements where the verb meaning has become semantically 'light' compared to the contribution of the noun to the meaning of the overall expression (Baldwin and Su Nam Kim, 2010). Particle and prepositional verbal constructions have one or more associated lexical items which modifiy the verb predicate, making the compositional analysis (literal meaning) unacceptable. As a particle verb, *se ut* means 'look' in the sense 'look like' or 'appear', while its literal meaning is 'look out', as in *se ut gjennom vinduet* 'look out (through) the window'.

Verbal constructions have been implemented as verbs with non-thematic objects (light verb constructions), as particle

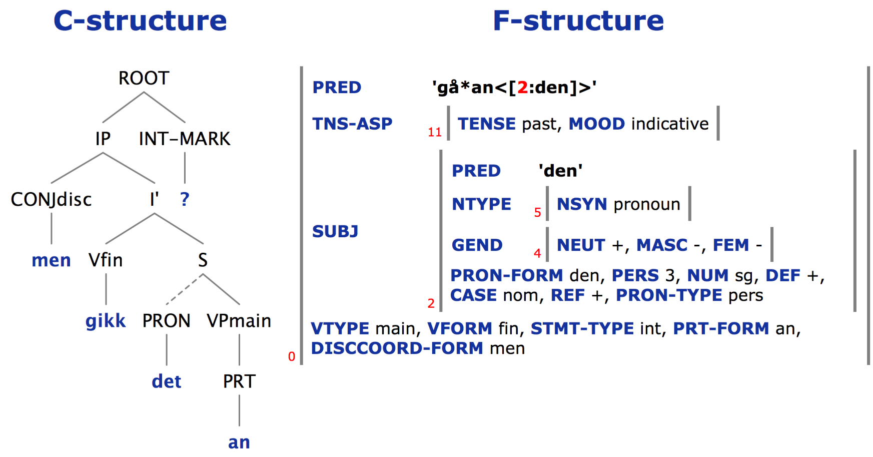


Figure 6: Constituent and functional structure representations with particle verb

verbs or as verbs with selected prepositions, depending on their syntactic properties. The LFG distinction between c- and f-structure allows NorGram to capture the compositional and non-compositional properties of MWEs in a perspicuous way. Thus, all the verb frame MWE types mentioned above express the non-compositional meaning of the MWE as augmented predicates in the predicate-argument structure: ‘ta*slutt<(↑SUBJ)>(↑OBJ)’, ‘se*ut<(↑SUBJ)>’, ‘vite*om<(↑SUBJ)(↑OBL-TH)>’. The words *slutt*, *ut* and *om* are analyzed respectively as a non-thematic object (outside the argument frame) selected by the verb entry, a selected particle, and a selected preposition. The c-structure captures the regular syntactically productive analysis of the expressions, allowing the usual variations in constituent order. Only when the MWEs allow no (or insignificant) formal variation and no intervening words may they be analyzed as ‘words with spaces’ in the lexicon.

An example of a particle verb from the Sofie corpus is provided in 6; the particle *an* is only found in MWEs.

(6) *Men det gikk an.*
but it went PRT
‘But it was possible.’

The MWEhood of the particle verb construction is shown by the complex predicate name in the value of the PRED feature in the f-structure in Figure 6. Examples of ‘words with spaces’ are expressions such as *med ett* ‘suddenly’ and *borte vekk* ‘gone’, which are fixed and thus simply added to the lexicon with the appropriate lexical category.

We also recorded problems with MWEs that belong to the group of semi-fixed expressions. As an example, the annotators suggested that the multiword unit *et eller annet* ‘some, something’ (literally ‘one or another’) was the main problem in the unsatisfactory analyses of the sentences in 7 and 8.

(7) *Altså måtte verdensrommet en eller annen*
thus must space:the one or another
gang ha blitt til av noe annet.
time have become to of something else
‘So space must once have been created from something else.’

(8) *Til syvende og sist måtte et eller annet*
to seventh and last must one or other
en gang ha blitt til av null og
one time have become to of zero and
niks.
nought
‘Ultimately something must once have been created from diddly-squat.’

The expression was added to the lexicon as a MWE quantifier (Q). Like many quantifiers in Norwegian, the MWE agrees with the quantified nominal, and two of its components — the determiner *en* and the adjectival determiner *annen* — inflect in gender. This was implemented by adding three different entries to the lexicon, one for each gender (*en eller annen* M, *ei eller anna F*, *et eller annet* NEUT). Adding one entry in the lexicon for each possible form of a MWE is not very economical. If we adopt this practice we risk overpopulation, and we fail to capture the morphological generalities of the set of inflectional forms, both common problems when treating MWEs as problems of the lexicon as described by Sag et al. (2002). These kinds of problems thus call for being somewhat restrictive with respect to adding new MWEs to the lexicon, and how to implement them. We want to capture as many MWEs as possible, but we also want to avoid representing every exception to the grammar rules in the lexicon. However, such constructions are so prevalent that they pose problems for syntactic analysis and thus cannot be ignored, amply illustrated by the sentence in 8. In the phrase structure tree representing its constituent structure, shown in Figure 7, the number of terminal nodes is far smaller than the number of individual words, reflecting the fact that there are four MWEs in the sentence.

MWE implementation accounts for 41% of the lexicon updates carried out after the first parse of Sofie. The main question remains how to represent flexibility, and how to account for families of constructions and other kinds of structural frames. Given the frequency of MWEs among the problems encountered, it seems reasonable to assume that the further grammar development would benefit from applying fixed criteria for MWEhood and from identifying

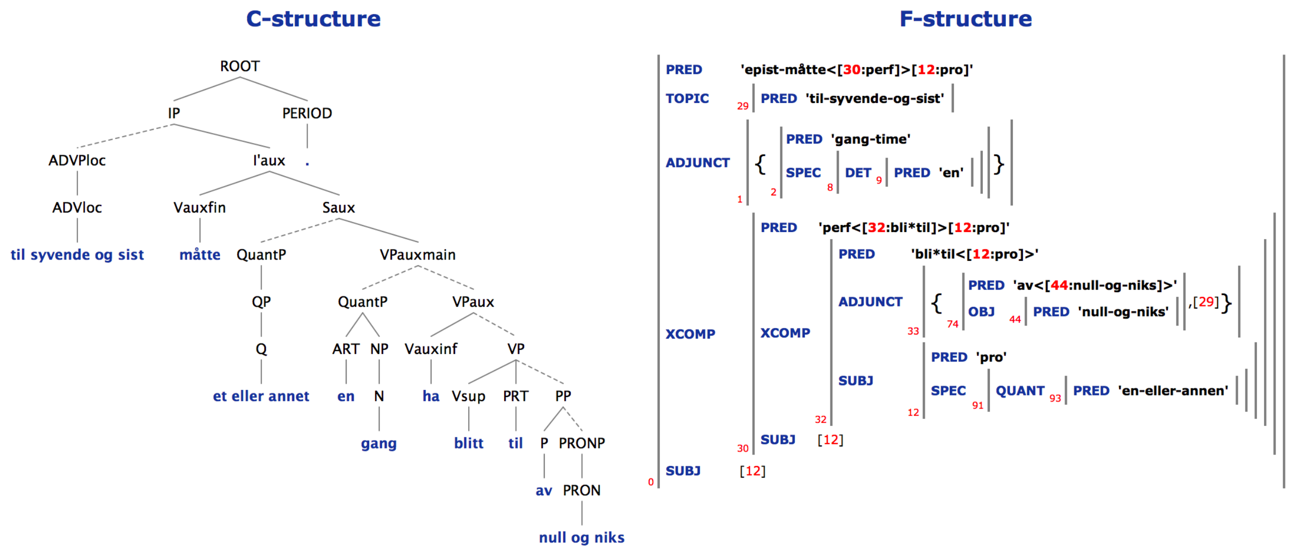

C-structure
F-structure
ROOT
IP
PERIOD
ADVPloc
I'aux
.
ADVloc
Vauxfin
Saux
til syvende og sist
måtte
QuantP
VPauxmain
QP
QuantP
VPaux
Q
ART
NP
Vauxinf
VP
et eller annet
en
N
ha
Vsup
PRT
PP
gang
blitt
til
P
PRONP
av
PRON
null og niks
PRED 'epist-måtte<[30:perf]>[12:pro]'
TOPIC 29 PRED 'til-syvende-og-sist'
ADJUNCT { PRED 'gang-time' SPEC DET PRED 'en' }
1 2 8 9
XCOMP PRED 'perf<[32:bli*til]>[12:pro]'
XCOMP PRED 'bli*til<[12:pro]>'
ADJUNCT { PRED 'av<[44:null-og-niks]>' OBJ 44 PRED 'null-og-niks' ,[29] }
33 74
SUBJ PRED 'pro' SPEC QUANT 93 PRED 'en-eller-annen'
12 91
32
30 SUBJ [12]
0 SUBJ [12]

Figure 7: Sentence with multiple MWEs

optimal techniques of analysis for the implementation of different types of MWEs. This could provide a more consistent treatment of MWEs in the grammar, and make us better equipped to meet the challenges MWEs pose in automatic processing.

4.3. Other lexicon problems

Following MWEs, the major group among the recorded lexicon updates is lexical category updates, with 31 % of the reported problems. In addition, we found some problems related to lexical frames (or subcategorization), new lexicon entries, and new word senses added to the lexicon. Lexical category updates are cases where words were reclassified after a problem had been reported for the sentences in which they occurred. An example is *riktignok* 'true', 'indeed', which was previously classified as a verb phrase adverb (ADV) in the lexicon.

(9) *Et menasjeri var en samling av forskjellige*
a menagerie was a collection of different
dyr, og riktignok — Sofie var ganske
animals and indeed Sofie was quite
godt fornøyd med sin egen samling.
well content with her own collection
'A menagerie was a collection of different animals, and Sofie was indeed quite content with her own collection.'

The part of speech *adverb* is a large class which encompasses many words with quite different syntactic properties. The words traditionally described as adverbs have verb phrase adverb as their default classification in our lexicon. In the case of *riktignok*, it was the syntactic distribution of this category, as specified in the grammar rules, that was found to cause problems for the constituency analysis. The example illustrates that lexical category updates are often grammatically motivated in the sense that words with a certain classification in the lexicon sometimes turn out to have syntactic properties that single them out as a separate category. After annotator intervention, *riktignok* was additionally classified as a root adverb (ADVroot). Root adverbs like *riktignok* differ from other adverbs because they can form utterances on their own.

The lexical categories in NorGram are fundamentally based on syntactic distribution, and by parsing Sofie we discovered several instances of previously unseen syntactic behavior of different lexical categories, or members of these categories, leading to reclassification in the lexicon. This may be further exemplified by the class of reflexive pronouns, which after intervention is divided into two categories, non-referring and referring reflexives. A referring reflexive in Norwegian is the MWE *seg selv* in sentences such as *barna vasker seg selv* 'the children wash themselves', as opposed to the non-referring *seg* in *barna vasker seg* 'the children wash', where *seg* is used with the reflexive verb *vaske*. Previously, NorGram did not have an analysis for the special case of referring *seg*, as found in 10 and 11.

(10) *Straks Sofie hadde lukket porten*
immediately Sofie had closed gate:the
bak seg, åpnet hun konvolutten.
behind self opened she envelope:the
'As soon as Sofie had closed the gate behind her, she opened the envelope.'

(11) *Sofie skyflet katten ut på trappen og*
Sofie shoved cat:the out on stair:the and
lukket døren etter seg.
closed door:the after self
'Sofie shoved the cat out onto the stairs and closed the door behind her.'

The examples show that the simple form *seg* can also be a referring reflexive in certain contexts, and due to these findings, this case is now singled out as a special category, PRONrfl2, restricted by the grammar to the relevant contexts.

The next type of lexicon update is the one we termed 'lexical frames'. These may also be defined as grammatically motivated updates, for instance when a word does not have the needed valency. An example is the verb *huske* 'remember', which was found to lack an intransitive reading. As a result, intransitive *huske-remember* has now been added to the lexicon. We also encountered an unacceptable analysis of a sentence with the MWE candidate *komme rekende på en fjøl*.

(12) *Det hadde bare kommet rekende på en fjøl.*
it had only come drifting on a board
'It had just appeared from nowhere.'

Instead of treating the whole expression as a MWE, we accounted for the construction *komme rekende*, which is an instance of a general infinitive–present participle construction restricted to certain verbs, and which was already implemented. The necessary update was made in the lexical entry for the verb *reke*.

As a result of the problem diagnostics, four new interjection-like words were also added to the lexicon as root adverbs (ADVroot). These were the words *næh* 'nah', *neivel* 'no indeed', *pøh* 'bah', and *fillern* 'darn'. Like most interjections, these are all very colloquial with a fairly unorthodox orthography.

The final type of lexicon update is the addition of new senses to words that are already in the lexicon. We recorded two cases of missing senses, of which two concerned the same word, the noun *gang* 'corridor'. One instance was found in the sentence *I neste øyeblikk var hun ute i gangen* 'The next moment she was out in the hallway'. The main problem was actually not the sense in itself, but restrictions on semantic features associated with that particular sense, since the grammar sometimes requires that semantic features must be checked against features of associated words. During the first parse, *gang* was only implemented as a temporal noun in the lexicon. After update, the lexicon now distinguishes between *gang-time* and *gang-corridor*. This also allows using the 'regular', non-temporal preposition *i* 'in' together with *gang*, something which was not possible after the first parse and which was the direct cause of the unintended initial analysis. Since the only implemented analysis of *gang* was temporal, only *i* with temporal features could be used with this noun. Having added the non-temporal *gang-corridor*, we are now able to choose the non-temporal *i*.

Among the problems identified as lexical updates, many proved to be problems also of the grammar. This demonstrates that the different components of a (computational) grammar are closely interrelated, and that updates to one of the components will almost certainly affect the other. We have several times experienced unfortunate outcomes of grammar and lexicon updates, and keep in mind that any change to the grammar bears a risk of allowing for overgeneralization and other unwanted side-effects.

4.4. Related work

Some related research has been aimed at analyzing causes of missing parser coverage and improving coverage by automatic lexical acquisition. Nicholson et al. (2008) give a breakdown of gaps in coverage, including 7% due to missing MWEs, and propose the addition of lexical entries by hypothesizing their types. In a similar vein, Villavicencio et al. (2007) cite 8% parse failures due to missing MWEs and propose a lexical type predictor as well. Other lexical type predictors are proposed by Cholakov and van Noord (2010), who assign lexical types to single words only, and Zhang et al. (2010), who take a step towards acquiring MWEs, although with limits on the valency and complexity of the covered constructions. Some of these methods seem compatible with our parse methods, but so far it seems that accurate remedial actions for MWEs still need a manual intervention step.

5. Conclusion

Treebanking by the automatic parsing of corpora is a way of validating a computational grammar and lexicon, thereby identifying gaps in the computational treatment of a language. The lexicon for Norwegian used in INESS is quite extensive and in general provides excellent coverage. Existing resources for Norwegian — and also for other languages — are, however, quite lacking with respect to MWEs. The lexical resources for Norwegian contain a number of MWEs which were taken from traditional dictionaries or from NorKompLeks (Nordgård, 1998), but the latter was also based on dictionary examples, not on corpus study. Our experience shows that these are insufficient. Although our small study showed that there were two main types of lexicon problems, lexical category problems and problems with multiword expressions, these two categories differ in an essential way. The lexical category problems concern the category assigned to items already within the lexicon. Solving these problems involves changing the lexical category. The big challenge posed by multiword expressions is that we do not have access to an inventory over them. Solving these problems is therefore an undertaking of a much greater dimension.

The interactive and iterative approach adopted in INESS makes it possible to integrate more and more MWEs into the lexicon and grammar and to thereby reach better coverage, especially in the semiautomatically disambiguated part of the treebank described in this paper. We will, however, also create a large parsebank that will be fully automatically disambiguated, and for this, we cannot rely on discovering MWEs during the annotation process.

The work done with lexical issues in INESS will contribute to improving the lexical resources available for Norwegian. But it is important that independent work is done on finding and analyzing MWEs, not only for Norwegian but also for other languages.

6. Acknowledgments

This research has received funding from the Research Council of Norway under the program National Financing Initiative for Research Infrastructure.

7. References

Timothy Baldwin and Su Nam Kim. 2010. Multiword expressions. In Nitin Indurkhya and Fred J. Damerau, editors, *Handbook of Natural Language Processing*, chapter 12. CRC Press, Boca Raton, FL, USA, second edition.

Emily M. Bender, Dan Flickinger, and Stephan Oepen. 2011. Grammar engineering and linguistic hypothesis testing: Computational support for complexity in syntactic analysis. In Emily M. Bender and Jennifer E. Arnold, editors, *Language from a Cognitive Perspective: Grammar, Usage, and Processing*, CSLI Lecture Notes 201, pages 5–29. CSLI Publications.

António Branco. 2009. LogicalFormBanks, the next generation of semantically annotated corpora: key issues in construction methodology. In Mieczyslaw Klopotek, Adam Przepiórkowski, Slawomir Wierzchón, and Krzysztof Trojanowski, editors, *Recent Advances in Intelligent Information Systems*, pages 3–12. Academic Publishing House EXIT, Warsaw.

Miriam Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. 2002. The Parallel Grammar project. In *Proceedings of COLING-2002 Workshop on Grammar Engineering and Evaluation, Taipei, Taiwan*.

David Carter. 1997. The TreeBanker: A tool for supervised training of parsed corpora. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 598–603, Providence, Rhode Island.

Kostadin Cholakov and Gertjan van Noord. 2010. Acquisition of unknown word paradigms for large-scale grammars. In Chu-Ren Huang and Dan Jurafsky, editors, *Proceedings of the 23rd International Conference on Computational Linguistics, Beijing*, volume Posters of *COLING '10*, pages 153–161, Stroudsburg, PA, USA. Association for Computational Linguistics.

Helge Dyvik. 2000. Nødvendige noder i norsk. Grunntrekk i en leksikalsk-funksjonell beskrivelse av norsk syntaks [Necessary nodes in Norwegian. Basic features of a lexical-functional description of Norwegian syntax]. In Øivin Andersen, Kjersti Fløttum, and Torodd Kinn, editors, *Menneske, språk og felleskap*. Novus forlag.

Jostein Gaarder. 1991. *Sofies verden: roman om filosofiens historie*. Aschehoug, Oslo, Norway.

John Maxwell and Ronald M. Kaplan. 1993. The interface between phrasal and functional constraints. *Computational Linguistics*, 19(4):571–589.

Jeremy Nicholson, Valia Kordoni, Yi Zhang, Timothy Baldwin, and Rebecca Dridan. 2008. Evaluating and extending the coverage of HPSG grammars: A case study for German. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odjik, Stelios Piperidis, and Daniel Tapias, editors, *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

Torbjørn Nordgård. 1998. Norwegian computational lexicon (NorKompLeks). In *Proceedings of the 11th Nordic Conference on Computational Linguistics (NoDaLiDa), Copenhagen*.

Stephan Oepen, Dan Flickinger, Kristina Toutanova, and Christopher D. Manning. 2004. LinGO Redwoods, a rich and dynamic treebank for HPSG. *Research on Language & Computation*, 2(4):575–596, December.

Victoria Rosén and Koenraad De Smedt. 2007. Theoretically motivated treebank coverage. In *Proceedings of the 16th Nordic Conference of Computational Linguistics (NoDaLiDa-2007)*, pages 152–159. Tartu University Library, Tartu.

Victoria Rosén, Paul Meurer, and Koenraad De Smedt. 2005. Constructing a parsed corpus with a large LFG grammar. In *Proceedings of LFG'05*, pages 371–387. CSLI Publications.

Victoria Rosén, Paul Meurer, and Koenraad De Smedt. 2007. Designing and implementing discriminants for LFG grammars. In Tracy Holloway King and Miriam Butt, editors, *The Proceedings of the LFG '07 Conference*, pages 397–417. CSLI Publications, Stanford.

Victoria Rosén, Paul Meurer, and Koenraad De Smedt. 2009. LFG Parsebanker: A toolkit for building and searching a treebank as a parsed corpus. In Frank Van Eynde, Anette Frank, Gertjan van Noord, and Koenraad De Smedt, editors, *Proceedings of the Seventh International Workshop on Treebanks and Linguistic Theories (TLT7)*, pages 127–133, Utrecht. LOT.

Ivan Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In *Lecture Notes In Computer Science. Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing*, volume 2276, pages 189–206. Springer.

Aline Villavicencio, Valia Kordoni, Yi Zhang, Marco Idiart, and Carlos Ramisch. 2007. Validation and evaluation of automatically acquired multiword expressions for grammar engineering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1034–1043, Prague, Czech Republic, June. Association for Computational Linguistics.

Yi Zhang, Timothy Baldwin, Valia Kordoni, David Martinez, and Jeremy Nicholson. 2010. Chart mining-based lexical acquisition with precision grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 10–18, Los Angeles, California, June. Association for Computational Linguistics.